Inventory Routing Investigations

A Thesis
Presented to
The Academic Faculty

by

Jin-Hwa Song

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Industrial and Systems Engineering
Georgia Institute of Technology
June 2004

Inventory Routing Investigations

Approved by:

Martin W. P. Savelsbergh, Committee Chair

George L. Nemhauser

Laurie Dutton
(Praxair, Inc.)

Chelsea C. White III

Alan Erera

Date Approved: 24 June 2004

To my wife, for her love and patience.

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to Dr. Martin Savelsbergh for serving as my thesis advisor, and providing invaluable advice and support. There is no way that I could have completed this thesis without his excellent insights, invaluable suggestions, and endless encouragement.

My research would not have been possible without the financial support of Praxair, Inc. I would like to especially thank Laurie Dutton for being so supportive and also being on my committee. I am grateful to Dr. Alan Erera, Dr. George L. Nemhauser, and Dr. Chelsea C. White III for serving as members of my committee.

Words are insufficient to express how much I owe to my families. My parents have always been there to encourage me. They have never doubted that I could do anything. Their endless love is the source of my courage. I want to thank my parents-in-law for their support and encouragement. My many thanks also to my sisters, sister-in-law, brothers-in-law, and nieces for their love. I owe special thanks to my wife, Se-kyoung Oh. Her support, patience, and love enabled me to complete this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The elimination of distribution inefficiencies, occurring due to the timing of customers' orders is an important reason for companies to introduce vendor managed inventory programs. By managing their customers' inventories, suppliers may be able to reduce demand variability and therefore distribution costs. We develop technology to measure the effectiveness of distribution strategies. Popular practical performance measures, such as volume delivered per mile, are effective in measuring relative performance, but inadequate to measure absolute performance. We develop a methodology that allows the computation of tight lower bounds on the total mileage required to satisfy customer demand over a period of time. As a result, companies will be able to gain insight into the effectiveness of their distribution strategy. This technology can also be used to suggest desirable delivery patterns and to analyze tactical and strategic decisions.

Secondly, we study a variant of the inventory routing problem, which we will refer to here as the inventory routing problem with continuous moves (IRP-CM). The typical inventory routing problem deals with the repeated distribution of a single product, from a single facility, with an unlimited supply, to a set of customers that can all be reached with out-and-back trips. Unfortunately, this is not always the reality. We introduce the IRP-CM to study two important real-life complexities: limited product availabilities at facilities and customers that cannot be served using out-and-back tours. We need to design delivery tours spanning several days, covering huge geographic areas, and involving product pickups at different facilities. We develop a heuristic and an optimization algorithm to construct distribution plans in environments where continuous moves are employed. The heuristic is an innovative randomized greedy algorithm, which includes linear programming based postprocessing technology. We demonstrate its effectiveness in an extensive computational study. To solve the IRP-CM to optimality, we give a time-discretized integer programming

model and develop a branch-and-cut algorithm. As instances of time-discretized models tend to be large we discuss several possibilities for reducing the problem size. We introduce a set of valid inequalities, called delivery cover inequalities, in order to tighten the bounds given by the LP relaxation of the time-discretized model. This class of delivery cover inequalities can be used more generally in vehicle flow models. In order to identify valid inequalities that are violated by the solution to the LP relaxation of the time-discretized model, we develop two separation algorithms. We also introduce branching schemes exploiting the underlying structure of the IRP-CM. An extensive computational study demonstrates the effectiveness of the optimization algorithm. Finally, we present an integrated approach using heuristics and optimization algorithms providing effective and efficient technology for solving inventory problems with continuous moves.

CHAPTER I

INTRODUCTION

In many industries, vendor managed inventory resupply (VMI) has become a popular strategy for reducing inventory holding and/or distribution costs. Examples of these industries include: the industrial gas industry, the petrochemical industry, the soft drink industry, the automotive industry, and the grocery industry.

In environments where VMI partnerships are being implemented, the vendor is allowed to choose both the timing and size of the deliveries. In exchange for this freedom, the vendor agrees to ensure that its customers will not run out of product. In a more traditional relationship, where customers call in their orders, large inefficiencies can occur due to the timing of the customers' orders, i.e., high inventory and high distribution costs. By employing VMI partnerships, companies may be able to reduce demand variability and, therefore, their inventory holding and distribution costs. However, realizing the cost savings opportunities of VMI partnerships is not an easy task, particularly with a large number and variety of customers. The inventory routing problem (IRP) seeks to do exactly that: determining a distribution strategy that minimizes long term distribution costs.

Our research is motivated by the work done with Praxair (www.praxair.com). Praxair is a leading global company that supplies industrial gases as well as related services and technologies to various industries including food, beverage, healthcare, semiconductor, chemical, refining, primary metal and metal fabrication. Atmospheric gases, such as Oxygen, Nitrogen, and Argon, are Praxair's primary products. These gases are delivered to customers in there liquid form by tank trucks. Since the trucks need to be product specific, the distribution problem of each gas is managed separately.

A reasonably large number of geographically dispersed plants produce Oxygen and Nitrogen. As a result, all Oxygen and Nitrogen customers can be reached with an out-and-back

trip from at least one of the plants. Consequently, a distribution environment has been created in which customers are assigned to a primary plant from which they receive (almost) all their product and in which each plant serves a set of customers using a set of drivers that make delivery tours visiting a few customers (typically not more than three) and that return back to the plant at the end of each tour. On the other hand, relatively few plants produce Argon and, as a result, a substantial portion of Argon customers cannot be reached by out-and-back tours from any of the plants. Therefore, a distribution environment has been created that relies on sleeper teams. A sleeper team consists of two drivers that can, in principle, work around the clock as they take turns driving (when one driver drives, the other driver sleeps). Sleeper teams are on the road for several days in a row, covering huge geographic areas, picking up product at different facilities along the way, and moving almost continuously. Another factor that necessitates the use of sleeper teams for Argon distribution is the fact that Argon is a co-product of Oxygen production. The rate at which Argon is produced depends on the demand for Oxygen rather than the demand for Argon. This leads to mismatches between product availability and product demand.

The traditional inventory routing problem deals with the repeated distribution of a single product, from a single production facility, to a set of customers with a fleet of homogeneous vehicles, over a certain planning horizon. Even in situations with multiple production facilities serving a certain set of customers, the customers are usually aligned with a particular production facility which results in the need to solve several single facility inventory routing problems. This is exactly Praxair's distribution environment for Oxygen and Nitrogen. A reasonably large body of literature exists on solution approaches for the inventory routing problem, and Praxair has incorporated many of these ideas into their planning tools. Unfortunately, Praxair has found that it is hard to judge the quality of the distribution plans produced. Praxair uses the volume delivered per mile to evaluate their distribution strategies. Although the volume delivered per mile by itself is not a meaningful number, because it is impacted by many factors, such as the geography of customer locations and customer usage patterns, it is valuable for comparing performance in consecutive periods of time. If a company has a stable customer set and the customer usage patterns do not

fluctuate significantly, then an increase in volume per mile indicates that the distribution planning is improving. The volume delivered per mile is a useful measure for monitoring relative distribution strategy performance. However, volume per mile cannot be used to determine, in an absolute sense, the quality of a distribution strategy.

In the first part of the thesis, we develop a methodology that allows the computation of tight lower bounds on the total mileage required to satisfy customer demand over a period of time (and thus upper bounds on volume per mile). This technology allows Praxair to gain insight into the effectiveness of their distribution strategy.

Unfortunately, there does not exist a large body of literature on inventory routing environments in which sleeper teams are employed, i.e., where delivery trucks are continuously moving. In fact, very little if any literature exists for that type of situation.

In the second part of the thesis, we formally introduce the inventory routing problem with continuous moves and develop solution approaches for this problem. The resulting technology can be used to construct effective distribution plans for Argon. Even though the research on continuous move distribution is motivated by Argon distribution as encountered by Praxair, there are many other distribution environments that can benefit from continuous move distribution. We end this section with some examples of the value of continuous moves.

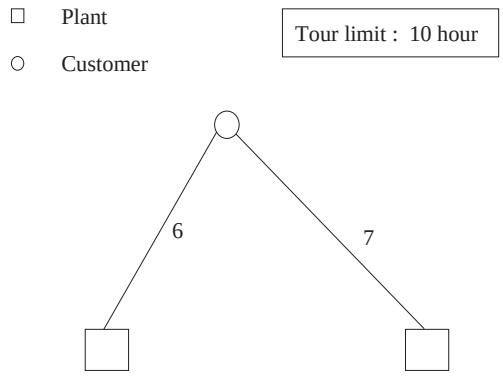


Figure 1: Necessity of Continuous Moves

In Figure 1, there are two plants and one customer. The tour limit is 10 hours and the

travel time from any plant to the customer is larger than 5 hours. The customer cannot be served without using continuous moves.

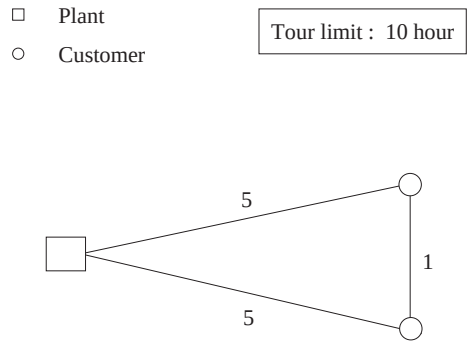


Figure 2: Cost effectiveness of Continuous Moves

Figure 2 shows the cost effectiveness of continuous moves. There are two customers and one plant. Assume that daily deliveries are necessary for the two customers and a vehicle can carry the sum of the delivery amounts for these two customers. Two vehicles need to perform the deliveries required. Each tour length is 10 hours. Therefore, the total travel time is 20 hours. If a vendor uses continuous moves, a vehicle can perform the deliveries required and the tour length would be 11 hours. By employing continuous moves for the distribution, more cost effective distribution can be obtained.

Figure 3 shows similar effects when employing continuous moves with multiple facilities. The number associated with a customer is the amount of product required to be delivered daily. The tank capacity of a vehicle is 10. Without employing continuous moves, a vendor needs 3 vehicles and the total travel time would be 24 hours. If a vendor uses continuous moves, two vehicles can perform the deliveries and the total travel time will be 18 hours. By employing continuous moves for the distribution, a more cost effective distribution will be obtained again.

The following two examples indicate the value of the continuous moves when mismatches between product availability and product demand occur.

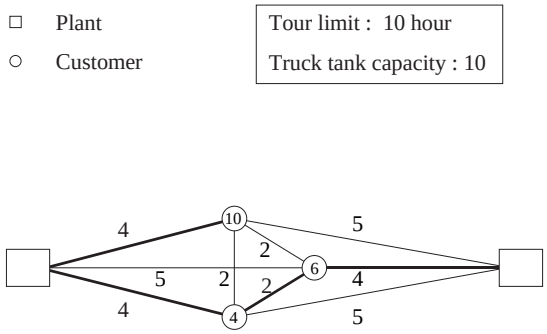


Figure 3: Cost effectiveness of Continuous Moves with multiple facilities

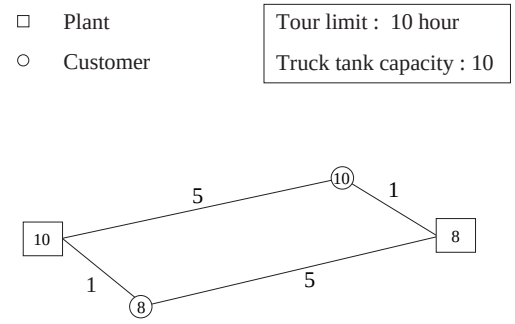


Figure 4: Cost effectiveness of Continuous Moves in Supply Driven Environments

In Figure 4, the number associated with the plant is its daily production rate. Without employing continuous moves for the product distribution, the total travel time would be 20 hours. With continuous moves, the total travel time would be 12 hours, because a vehicle does not need to return to its base.

In Figure 5, stockouts cannot be avoided without employing continuous moves. Supply driven environments, furthermore, exemplify situations in which vendors should consider employing continuous moves for their distribution systems. The examples above indicate

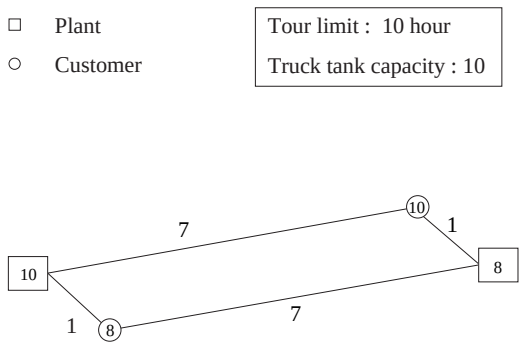


Figure 5: Necessity of Continuous Moves in Supply Driven Environments

that we can design more effective distribution strategies by employing continuous moves.

1.1 Literature Review

We discuss several papers representing a variety of solution approaches to the inventory routing problem. For an overview of the major research activities in this area, see Campbell et al. [12] and Kleywegt et al. [25].

The inventory routing problem is a long-term planning problem. Due to the difficulty of solving long-term planning problems, most approaches presented in the literature deal with a sequence of short-term problems. The short-term planning horizon in the earliest work on the inventory routing problem is a single day. This approach is implemented by Federgruen and Zipkin [20], Golden et al. [23], and Chien et al. [15].

Fisher et al. [21] and Bell et al. [9] use an integer program to decide which customers to visit in the next few days. In their approach, the integer programming model deals with a set of potential routes and decides delivery volumes for customers, assignment of customers to routes, assignment of vehicles to routes, and start times for each selected route. Campbell [11] and Campbell et al. [10] use a two-phase approach. They use an integer program to determine which customers to visit in phase I, and then use insertion heuristics to determine delivery routes and quantities in phase II.

Another group of researchers use an analysis of a single customer to determine which customers to visit. Dror and Ball [18, 19] propose a way to consider the long-term effects of short-term decisions by analyzing each customer's optimal replenishment day t^*, the expected increase in future cost if the delivery is made on day t instead of t^*, and a future benefit of early delivery. Using ideas similar to Dror and Ball [18, 19], Bard et al. [8] and Jaillet et al. [24] consider satellite facilities where vehicles can be reloaded and customer deliveries can be continued until the closing time is reached. However, their problem is still a single depot problem and they assume that all customers can be visited on an out-and-back trip from the depot.

From a perspective of long-term planning problems, it is unlikely to know customers' demands precisely. Several researchers incorporate customer usage uncertainty into the inventory routing problem. This problem is called a stochastic inventory routing problem. This approach can be found in Minkoff [28], Kleywegt et al. [25, 26], Nori [30], and Adelman[2, 3].

Gallego and Simchi-Levi [22], Anily and Federgruen [5, 6] and Webb and Larson [33] study the inventory routing problem over an infinite horizon. Gallego and Simchi-Levi present a lower bound on the long run average cost over all inventory-routing strategies. This lower bound is similar to our simple lower bound presented in Section 2.2. By using this lower bound, they show that direct shipping is at least 94% effective whenever the minimum of the economic lot size is at least 71% of the vehicle capacity. Not surprisingly, the effectiveness deteriorates as economic lot sizes get smaller. Anily and Federgruen minimize long run average transportation and inventory costs by determining long-term routing patterns. They consider a class of strategies in which a collection of regions (sets of customers) is specified. If a customer appears in more than one region, they assign a specific fraction of its demand to each of these regions. Whenever a customer in a region receives a delivery, all other customers in that region are also visited in an efficient route. They use a modified circular regional partitioning procedure to determine long-term routing patterns. Webb and Larson [33] discuss the minimum fleet size required to serve the customers. They separate customers into disjoint clusters and additionally create a route sequence for each cluster

using a savings approach that minimizes vehicle utilization, which minimizes the number of vehicles effectively. Chan, Federgruen, and Simchi-Levi [14] identify a combined inventory policy and a routing strategy to minimize cost over an infinite time horizon. They show the asymptotic effectiveness of the class of fixed partition policies as well as those employing zero inventory ordering. They provide worst case and probabilistic bounds under certain assumptions.

As shown above, the inventory routing problem has been studied in various ways by many researchers. However, to our knowledge, the variant with continuous moves have never been considered in the inventory routing literature. On the other hand, the concept of continuous moves has been studied in other contexts, e.g., ship routing problems. For a recent overview of the major research activities in the ship routing and scheduling problem, see Christiansen et al. [17]. Christiansen [16] studies a real ship planning problem, in which she combines an inventory management problem and a ship routing problem. Using the information about the production and consumption pattern of ammonia, she develops technology to design routes for a fleet of ships transporting ammonia from production harbors to consumption harbors. She gives an IP formulation and uses a Dantzig-Wolfe decomposition approach to solve the problem. A branch-and-bound is used to solve the whole problem. Since a ship visits multiple supply harbors as well as multiple demand harbors, the problem introduced by Christiansen is similar to the IRP-CM. However, it is unlikely that the solution approach proposed by Christiansen can be used to solve the IRP-CM because the ship routing problem only deals with a relatively small number of harbors and the specialized solution techniques developed do not seem applicable to instances with even a moderate number of harbors.

In the last part of this thesis we develop a branch-and-cut algorithm for the IRP-CM. Branch-and-cut has been very successful in solving large instances of the traveling salesman problem. Although the research effort spent on the capacitated vehicle routing problem with branch-and-cut cannot be compared with that for the traveling salesman problem, branch-and-cut is the most promising solution method for the capacitated vehicle routing problem at present. Naddef and Rinaldi [29] give an overview of the research activities

in branch-and-cut algorithms for the capacitated vehicle routing problem. This literature is relevant to our research as the IRP-CM has many characteristics also encountered in capacitated vehicle routing problems. For recent results, see Ralphs et al. [31], Achuthan et al. [1], and Lysgaard et al. [27]. We adopt some of ideas from Ralphs et al. [31] when we develop the branch-and-cut algorithm for the IRP-CM.

1.2 Contributions

In this section, we list the major contributions of this thesis.

- We develop technology that allows a vendor to measure the effectiveness of its distribution strategy in an absolute sense. This technology can also be used to suggest desirable delivery patterns and to analyze tactical and strategic decisions. These decisions include: minimum fleet size to serve customers, facility location problems, customer-plant alignments, capital investments to improve the efficiency of its distribution system, and so on.

- We develop technology that allows a vendor to construct distribution plans in environments where continuous moves are employed. This variant of the inventory routing problem, which is defined and analyzed in this thesis, captures both production and consumption complexities. We develop an innovative randomized greedy algorithm, which includes linear programming based postprocessing technology, to solve the problem.

- We also develop a branch-and-cut algorithm for its solution. A class of delivery cover inequalities is introduced for the branch-and-cut algorithm. We develop two heuristics to separate delivery cover inequalities violated by LP solution. Delivery cover inequalities can be used in other vehicle flow models.

PART I

Performance Measurement for Inventory Routing

CHAPTER II

PERFORMANCE MEASUREMENT FOR INVENTORY ROUTING

2.1 Introduction

In the first part of this thesis, we do not focus on developing distribution strategies, but instead on measuring the effectiveness of distribution strategies. A popular performance measure used in practice to evaluate distribution strategies in an environment where VMI partnerships are in effect is the volume delivered per mile or *volume per mile* for short. As the volume that needs to be delivered by the vendor over a given period of time is determined by the total usage of its customers, and not under the control of the vendor, the vendor strives to minimize the total mileage required to deliver product. However, volume per mile by itself is not a meaningful number, because it is impacted by many factors, such as the geography of customer locations and customer usage patterns, but it is valuable for comparing performance in consecutive periods of time. If a company has a stable customer set and customer usage patterns do not fluctuate much, then an increase (decrease) in volume per mile indicates that distribution planning is improving (worsening).

The above discussion shows that volume per mile is a useful measure for monitoring relative distribution strategy performance. However, volume per mile cannot be used to determine, in an absolute sense, the quality of a distribution strategy. We develop a methodology that allows the computation of tight lower bounds on the total mileage required to satisfy customer demand over a period of time (and thus upper bounds on volume per mile). As a result companies will be able to gain insight into the effectiveness of their distribution strategy.

The remainder of Part I is organized as follows. In Section 2.2 we present a simple bound on the minimum total mileage required to satisfy customer demand. In Section 2.3

we analyze two 2-customer examples. The analysis reveals a crucial insight that forms the basis for the methodology developed and discussed in Section 2.4. In Chapter 3, we present a variety of computational experiments conducted with real-life data to evaluate our proposed methodology. Finally, in Chapter 4, we discuss other potential uses of the technology developed.

2.2 A Simple Bound

Consider the following variant of the inventory routing problem. A single product has to be distributed from a single facility to a set I of n customers over a period of time of length T. Each customer $i \in I$ has the capability to maintain a local inventory of product up to a maximum of C_i. In the period of interest customer i consumes an amount u_i of product. A fleet of homogeneous vehicles, with capacity Q, is available for the distribution of the product. We assume an unlimited supply of product and an unlimited number of vehicles in the fleet. We denote the travel distance between two locations i and j by t_{ij}. The objective is to obtain an accurate estimate of the minimum total mileage required to satisfy customer demand. Observe that when $C_i \geq Q \ \forall i \in I$, then the optimal distribution strategy is to always deliver a full truck load to a customer right when the customer's storage tank becomes empty. The resulting total distance is $\sum_{i \in I} \frac{u_i}{Q} 2t_{0i}$, where 0 denotes the plant. Therefore, a simple lower bound on the minimum total mileage required to satisfy customer demand is obtained by simply assuming that all customers' storage capacities are greater than the truck capacity, i.e.,

$$\sum_{i \in I} \frac{u_i}{Q} 2t_{0i}.$$

This results in the following simple upper bound on volume per mile

$$\frac{\sum_{i \in I} u_i}{\sum_{i \in I} \frac{u_i}{Q} 2t_{0i}}.$$

2.3 Towards an Improved Bound

In practice, deliveries to customers with storage capacity less than the truck's capacity, i.e., $C_i < Q$, are usually combined with other deliveries to ensure a high utilization of the truck's capacity. The analysis of the two 2-customer examples presented below suggests how to

incorporate varying storage capacity at customers in the calculation of a lower bound on the minimum total mileage required to satisfy customer demand.

2.3.1 Example 1

Consider the distribution environment depicted in Figure 1, i.e., a single plant and two customers.

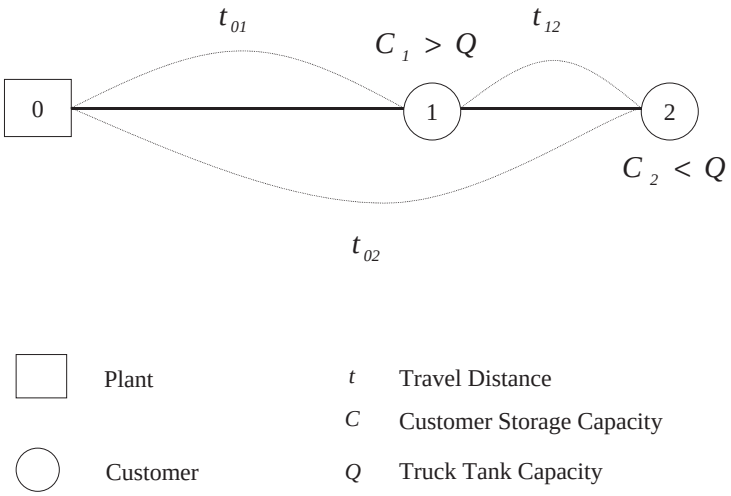


Figure 6: Two customer configuration of Example 1

In this example, $LB_1 = \frac{u_1}{Q}2t_{01} + \frac{u_2}{Q}2t_{02}$. Since $C_2 < Q$, whenever a truck goes to Customer 2 with full truck load, $Q - C_2$ of product is left in its tank. Note that $t_{02} = t_{01} + t_{12}$ in this example. The leftover product can be used to satisfy the need for product of Customer 1 at no extra cost. To deliver u_2 to Customer 2, at least $\frac{u_2}{C_2}$ deliveries have to be made. Therefore, at least $\frac{u_2}{C_2}(Q - C_2)$ leftover product is available for Customer 1. Two cases have to be considered: (1) the leftover product is sufficient to satisfy Customer 1's needs, and (2) the leftover product is insufficient to satisfy Customer 1's needs. Let D^* denote the best possible lower bound on the minimum total mileage required to satisfy customer demand.

Case 1 : If $\frac{u_2}{C_2}(Q - C_2) \geq u_1$, then

$$D^* = \frac{u_2}{C_2} 2t_{02}.$$

Case 2 : If $\frac{u_2}{C_2}(Q - C_2) < u_1$, then

$$D^* = \frac{u_2}{C_2} 2t_{02} + \frac{u_1 - \frac{u_2}{C_2}(Q - C_2)}{Q} 2t_{01}.$$

For Case 2, we sent full trucks to Customer 1 to satisfy the remaining product need, i.e., $u_1 - \frac{u_2}{C_2}(Q - C_2)$.

OBSERVATION 1. *The delivery patterns used in the best possible lower bound are among* $(Q, 0), (Q - C_2, C_2)$, *and* $(0, C_2)$.

2.3.2 Example 2

Consider the distribution environment depicted in Figure 2, i.e., a single plant and two customers. In this example, $LB_1 = \frac{u_1}{Q} 2t_{01} + \frac{u_2}{Q} 2t_{02}$. Since $C_1 < Q$, whenever a truck

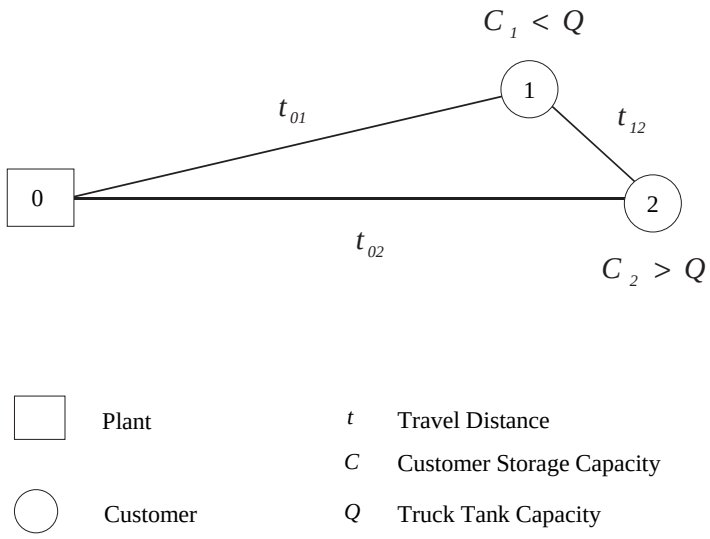


Figure 7: Two customer configuration of Example 2

goes to Customer 1 with full truck load, $Q - C_1$ of product is left in its tank. So if this leftover product is used to satisfy product need of Customer 2, $\frac{Q}{Q - C_1}$ trips with leftover

product are necessary to deliver Q. Whenever leftover product is delivered to Customer 2, $t_{12}+t_{02}-t_{01}$ additional miles are incurred. Therefore, the travel distance incurred to deliver Q to Customer 2 with leftover product from Customer 1 is $\frac{Q}{Q-C_1}(t_{12}+t_{02}-t_{01})$. The travel distance incurred to deliver Q to Customer 2 directly from the plant is $2t_{02}$. Consequently, if $\frac{Q}{Q-C_1}(t_{12}+t_{02}-t_{01}) < 2t_{02}$, it is better to use leftover product at Customer 1 to satisfy the product need of Customer 2. For the remainder, assume that this is the case, i.e., $\frac{Q}{Q-C_1}(t_{12}+t_{02}-t_{01}) < 2t_{02}$. Two cases have to be considered: (1) the leftover product is sufficient to satisfy Customer 2's needs, and (2) the leftover product is insufficient to satisfy Customer 2's needs.

Case 1 : If $\frac{u_1}{C_1}(Q-C_1) \geq u_2$, then

$$D^* = \frac{u_2}{Q-C_1}(t_{01}+t_{12}+t_{02}) + \frac{u_1 - \frac{u_2}{Q-C_1}C_1}{C_1}2t_{01}$$

Case 2 : $\frac{u_1}{C_1}(Q-C_1) < u_2$, then

$$D^* = \frac{u_1}{C_1}(t_{01}+t_{12}+t_{02}) + \frac{u_2 - \frac{u_1}{C_1}(Q-C_1)}{Q}2t_{02}.$$

OBSERVATION 2. *The only delivery patterns used in an optimal solution are among* $(C_1, 0)$, $(0, Q)$, *and* $(C_1, Q-C_1)$.

The two observations above form the basis for the methodology developed to compute improved bounds on the minimum total mileage required to satisfy customer demand.

2.4 An Improved Bound

Define a feasible delivery pattern $P_j = (d_{j1}, d_{j2}, ..., d_{jn})$ to be a delivery pattern that satisfies $\sum_{i \in I} d_{ji} \leq Q$ and $0 \leq d_{ji} \leq C_i \ \forall i \in I$. Let $\delta(P_j) = \{i \in I : d_{ji} > 0\}$ denote the set of customers visited in delivery pattern P_j. The cost of delivery pattern P_j, denoted as $c(P_j)$, is the value of an optimal solution to the traveling salesman problem involving the plant and the customers in $\delta(P_j)$. Let $\mathcal{P}$ be the set of all feasible delivery patterns and let x_j be a decision variable indicating how many times delivery pattern P_j is used. Then the optimal objective function value of the following linear program, called the pattern selection LP,

provides a lower bound on the total mileage required to satisfy customer demand

$$\begin{aligned} D^* = \quad & min \quad \sum_{j:P_j \in \mathcal{P}} c(P_j) x_j \\ & s.t. \quad \sum_{j:P_j \in \mathcal{P}} d_{ji} x_j \geq u_i, \quad \forall i \in I \\ & \qquad x_j \geq 0 \end{aligned}$$

There are two major obstacles to using this linear program:

- The number of feasible delivery patterns is prohibitively large.
- The calculation of the cost of each delivery pattern involves the solution of a traveling salesman problem.

In the remainder of this section we discuss how these obstacles can be handled in practice. (We will assume throughout that distances satisfy the triangle inequality.)

We will start by showing that a much smaller set of delivery patterns can be considered when solving the linear program (an insight resulting from the analysis presented in the previous section).

DEFINITION 1. *(Base Pattern) A feasible delivery pattern P is a base pattern if at most one customer, say k, in $\delta(P)$ receives a delivery quantity less than $min(C_k, Q)$, and, in that case, the delivery quantity is $Q - \sum_{i \in \delta(P) \setminus \{k\}} C_i$.*

The base patterns can be divided into two classes:

1. $\sum_{i \in \delta(P)} C_i \leq Q$ so that $d_i = C_i \ \forall i \in \delta(P)$, and
2. $\sum_{i \in \delta(P)} C_i > Q$ so that there exists one customer, say k, with $d_k = Q - \sum_{i \in \delta(P) \setminus \{k\}} C_i$ and $d_i = min(C_i, Q) \ \forall i \in \delta(P) \setminus \{k\}$.

THEOREM 1. *The base patterns are sufficient to find an optimal solution to the Pattern Selection LP.*

Proof. Case 1. A feasible pattern P with $\sum_{i\in\delta(P)} d_i < Q$.

Suppose $\sum_{i\in\delta(P)} C_i < Q$. Then there exits a base pattern P' with $\delta(P) = \delta(P')$ and $d'_i = C_i \ \forall i \in \delta(P)$. Because $c(P) = c(P')$ and $d_i \leq d'_i \ \forall i \in \delta(P)$, we can replace P in any optimal solution by P'. Suppose $\sum_{i\in\delta(P)} C_i > Q$. Then there exits a pattern P' with $\delta(P) = \delta(P')$, $d'_i \geq d_i; \forall i \in \delta(P)$, and $\sum_{i\in\delta(P')} d'_i = Q$. Because $c(P) = c(P')$ and $d_i \leq d'_i \ \forall i \in \delta(P)$, we can replace P in any optimal solution by P'. Such patterns are covered in Case 2.

Case 2. A feasible pattern P with $\sum_{i\in\delta(P)} d_i = Q$.

We will show that such a pattern can be represented by a convex combination of base patterns with $\delta(\cdot) \subseteq \delta(P)$ (which implies that $c(\cdot) \leq c(P)$). Consider a feasible pattern P with $\sum_{i\in\delta(P)} d_i = Q$. Without loss of generality, assume that $\delta(P) = \{1, 2, ..., m\}$. Let $\{P_1, P_2, ..., P_{n_p}\}$ be the set of base patterns with $\delta(\cdot) \subseteq \delta(P)$. Let A be the $(m+1)\times n_p$ matrix in which the j^{th} column is $(d_{j1}, d_{j2}, ..., d_{jm}, 1)^T$. (Where d_{ji} is pattern P_j's i^{th} element.) Let $b^T = (d_1, d_2, ..., d_m, 1)$. We have to show that the linear system $Ax = b, \ x \geq 0$ has a feasible solution. We do so using Farkas' Lemma.

Farkas' Lemma. *A linear system $Ax = b, \ x \geq 0$ has a feasible solution if and only if $yb \geq 0$ for each y with $yA \geq 0$.*

Suppose $yb < 0$ for some y with $yA \geq 0$. Without loss of generality, we assume $y_1 \geq y_2 \geq \cdots \geq y_m$. Then there exists a base pattern P' such that $d'_i = min(C_i, Q) \ \forall i \in \{l+1, l+2, ..., m\}$, $d'_l = Q - \sum_{i=l+1}^{m} min(C_i, Q)$, and $d'_i = 0, \ \forall i \in \{1, 2, ..., l-1\}$. Since $(d'_1, d'_2, ..., d'_m, 1)^T$ is a column of A, we have $\sum_{i=1}^{m} d'_i y_i + y_{m+1} \geq 0$. Now, since $\sum_{i=1}^{m} d_i = \sum_{i=1}^{m} d'_i = Q$ and P is a feasible pattern, $\sum_{i=k}^{m} d_i \leq \sum_{i=k}^{m} d'_i \ \forall k \in \{1, 2, ..., m\}$. Therefore, $\sum_{i=1}^{k} d_i \geq \sum_{i=1}^{k} d'_i \ \forall k \in \{1, 2, ..., m\}$ and thus $\sum_{i=1}^{k}(d_i - d'_i) \geq 0 \ \forall k \in \{1, 2, ..., m-1\}$ and $\sum_{i=1}^{m}(d_i - d'_i) = 0$. Since $y_i \geq y_{i+1}$, we have $\sum_{i=1}^{k}(d_i - d'_i)y_k \geq \sum_{i=1}^{k}(d_i - d'_i)y_{k+1} \ \forall k \in \{1, 2, ..., m-1\}$. Since $\sum_{i=1}^{m}(d_i - d'_i) = 0$, we have $\sum_{i=1}^{m}(d_i - d'_i)y_m = 0$. By summing these m inequalities, we obtain $\sum_{i=1}^{m}(d_i - d'_i)y_i \geq 0$. But then $\sum_{i=1}^{m} d'_i y_i + y_{m+1} \leq \sum_{i=1}^{m} d_i y_i + y_{m+1} = yb < 0$, a contradiction. □

Now that we have significantly reduced the number of delivery patterns, we turn our attention to the number of customers visited in a delivery pattern as that impacts the effort required to compute the cost of a delivery pattern.

For any natural number k, let $C'_i = \frac{Q}{k}$ if $C_i < \frac{Q}{k}$ and $C'_i = C_i$ if $C_i \geq \frac{Q}{k}$. Observe that with these modified storage capacities a base pattern contains at most k customers. Let LB_k denote the optimal value of the pattern selection LP with base patterns based on the modified storage capacities. It is easy to see that LB_k provides a lower bound on D^* for every k and that $LB_1 \leq LB_2 \leq LB_3 \leq ....$ Finally, when $\frac{Q}{k} \leq min\{C_1, C_2, ..., C_n\}$, then $LB_k = D^*$. Note that the simple bound discussed in Section 2.2 is equal to LB_1.

For any natural number k, we can also compute an upper bound UB_k on D^* as follows. We let UB_k be the optimal objective function value of the pattern selection LP in which we only consider base patterns with at most k customers. It is easy to see that $UB_1 \geq UB_2 \geq UB_3 \geq ...$ and that when $k \geq \left\lceil \frac{Q}{min\{C_1, C_2, ..., C_n\}} \right\rceil$, then $UB_k = D^*$.

Our computational experiments have shown that tight bounds on D^* are already obtained for values $k = 3$ and $k = 4$, in the sense that the gap between LB_k and UB_k is very small (for the data sets of interest to us). Furthermore, for values $k = 3$ and $k = 4$, the traveling salesman problems that have to be solved involve at most 4 and 5 cities, respectively, and thus can be solved relatively easily by enumeration.

Our initial computational experiments have also shown that even though we have significantly reduced the number of delivery patterns in the pattern selection LP by restricting ourselves to base patterns, as the number of customers increases - especially the number of customers whose storage capacities are small - the number of base patterns increases rapidly. For example, for one of our larger instances with about 200 customers 22,575,528 base patterns were generated to compute UB_4. Even when using carefully designed memory-efficient implementations, the memory requirements become excessive. To be able to handle such large instances (and even larger ones) effectively, we have developed two additional techniques.

So far, we have only exploited feasibility considerations to reduce the set of delivery patterns that need to be considered. Next, we will show how optimality considerations can

be exploited effectively to reduce the set of delivery patterns that need to be considered. Consider a base pattern $P = \{d_1, d_2, \ldots, d_n\}$, all base patterns P_j visiting a subset of the customers in $\delta(P)$, and the following linear program, called the dominance LP,

$$\begin{aligned} z = \quad min \quad & \sum_{\{j:\delta(P_j)\subsetneq\delta(P)\}} c(P_j)\lambda_j \\ s.t. \quad & \sum_{\{j:\delta(P_j)\subsetneq\delta(P)\}} d_{ji}\lambda_j \geq d_i, \quad \forall i \in \delta(P) \\ & \lambda_j \geq 0 \end{aligned}$$

If $z \leq c(P)$, then the base patterns with $\lambda_j > 0$ collectively dominate the base pattern P and base pattern P can be eliminated from the pattern selection LP. However, even though the size of a dominance LP is small, setting up and solving it for every base pattern to determine if the base pattern is dominated is computationally prohibitive. Therefore, we rely on easily computable upper bounds on the optimal value of a dominance LP for dominance testing; if $z \leq z_{UB} \leq c(P)$, where z_{UB} denotes an upper bound on z, then the base pattern P is dominated and can be eliminated. We compute upper bound z_{UB} by restricting our attention to carefully selected subsets of patterns.

To illustrate, consider a base pattern $P = \{C_1, C_2, C_3, d_4\}$ with $d_4 < C_4$. (Note that this implies that $C_1 + C_2 + C_3 + d_4 = Q$.) If one of the following three conditions is satisfied for P, then P is dominated:

Condition 1: $c(P) \geq c(P_{123}) + \frac{d_4}{min\{Q,C_4\}}c(P_4)$ where $P_{123} = \{C_1, C_2, C_3, 0\}$ and $P_4 = \{0, 0, 0, min\{Q, C_4\}\}$.

Condition 2: $c(P) \geq \left(1 - \frac{d_4}{d_{14}+d_{24}+d_{34}}\right) c(P_{123}) + \frac{d_4}{d_{14}+d_{24}+d_{34}} \left(c(P_{14}) + c(P_{24}) + c(P_{34})\right)$ where $d_{14} = min\{Q - C_1, C_4\}$, $d_{24} = min\{Q - C_2, C_4\}$, $d_{34} = min\{Q - C_3, C_4\}$, $P_{123} = \{C_1, C_2, C_3, 0\}$, $P_{14} = \{C_1, 0, 0, d_{14}\}$, $P_{24} = \{0, C_2, 0, d_{24}\}$, and $P_{34} = \{0, 0, C_3, d_{34}\}$.

Condition 3: $c(P) \geq \left(1 - \frac{2d_4}{d_{124}+d_{134}+d_{234}}\right) c(P_{123}) + \frac{d_4}{d_{124}+d_{134}+d_{234}} \left(c(P_{124}) + c(P_{134}) + c(P_{234})\right)$ where $d_{124} = min\{Q - C_1 - C_2, C_4\}$, $d_{134} = min\{Q - C_1 - C_3, C_4\}$, $d_{234} = min\{Q - C_2 - C_3, C_4\}$, $P_{123} = \{C_1, C_2, C_3, 0\}$, $P_{124} = \{C_1, C_2, 0, d_{124}\}$, $P_{134} = \{C_1, 0, C_3, d_{134}\}$, and $P_{234} = \{0, C_2, C_3, d_{234}\}$.

The effectiveness of these simple dominance tests is demonstrated by the results presented in Table 1. The table shows the number of base patterns before and after applying the dominance tests for some of our larger instances.

Table 1: Effect of dominance tests

Instance	before	after
1	5,015,046	3,029,980
2	4,488,526	3,395,575
3	7,665,722	4,336,466
4	7,579,201	5,381,540
5	9,086,385	5,420,907
6	15,180,701	8,838,137
7	14,471,228	8,975,615
8	22,575,528	16,640,122

Next, we observe that a pattern selection LP has a large aspect ratio, i.e., a large ratio of number of columns to number of rows. Linear programs with large aspect ratios occur frequently when set partition or set covering formulations are used to model practical situations, for example in air crew scheduling applications. Specialized linear programming solvers exploiting the fact that most variables will have a zero value in an optimal solution have been developed for such problems. In the CPLEX linear optimization system, the specialized linear programming solver for high aspect ratio linear programs is the sifting optimizer. The sifting optimizer solves an LP with only a subset of the variables (assuming a zero solution value for each of the remaining variables). From the solution to this partial LP, the reduced costs of the remaining variables can be computed. Variables with reduced costs less than zero are added to the partial LP, the partial LP is resolved, and the process repeats. If no negative reduced cost variables exist, then the current solution is an optimal solution to the full problem. (This approach was first introduced by IBM under the name SPRINT approach [4].) Table 2 shows a comparison of cpu times for the CPLEX linear optimizer and the CPLEX sifting optimizer on some of our larger instances of the pattern selection LP.

In Table 2, we present the number of patterns generated, the number of major iterations

Table 2: Effect of sifting optimizer

Instance	# of patterns	# of iterations	default(sec)	sifting(sec)
1	3,029,980	5	85.66	77.09
2	3,395,575	5	167.15	94.07
3	4,336,466	6	118.37	121.34
4	5,381,540	6	308.44	244.90
5	5,420,907	6	155.29	152.56
6	8,838,137	5	360.20	240.83
7	8,975,615	6	397.33	254.81
8	16,640,122	6	675.98	533.56

of the sifting optimizer, and the cpu time taken by the default and sifting optimizer.

Using CPLEX' sifting optimizer does not resolve our memory issues as all delivery patterns still need to be loaded into memory. Therefore, we have developed our own implementation of a sifting optimizer. During the pattern generation phase, we do not load all base patterns into the linear programming solver, but only a subset of patterns that fits into memory and is highly likely to include an optimal solution. After solving this partial LP, we execute the pattern generation phase again, but this time we evaluate the reduced costs of the patterns (as opposed to their regular cost) and add patterns with a negative reduced cost to the partial LP. The algorithm terminates when no patterns can be added in an iteration.

In implementing this approach, we have to take into account that our pattern generation phase is computationally intensive as it involves solving a, albeit small, TSP for every pattern. As a consequence, we have to strike a proper balance between pattern generation and memory management. The ultimate goal is to solve the pattern selection LP with only two passes through pattern generation, i.e., one to generate a partial pattern selection LP and one to verify that all patterns left out of the partial pattern selection LP have nonnegative reduced costs. We want to avoid having to go through more than two pattern generation passes. To achieve this goal, we generate patterns in a specific order, first patterns involving a stop at a single customer, second patterns involving stops at two customers, third patterns involving stops at three customers, and finally patterns involving

stops at four customers, and add patterns to the partial pattern selection LP in batches, solving the partial selection LP after each batch of patterns has been added. A more detailed description can be found in Algorithm 1.

Algorithm 1 Sifting Optimizer

```
generate := true
ρ3 := ...
ρ4 := ...
Generate all patterns of size 1 and 2 and add them to the partial pattern selection LP
Solve the partial pattern selection LP
while generate = true do
  generate := false
  Generate all patterns of size 3 and add those with reduced cost less than ρ3 to the
  partial pattern selection LP
  ρ3 = 0
  if patterns were added to partial selection LP then
    generate := true
    Solve the partial pattern selection LP
  end if
  Generate all patterns of size 4 and add those with reduced cost less than ρ4 to the
  partial pattern selection LP
  ρ4 = 0
  if patterns were added to partial selection LP then
    generate := true
    Solve the partial pattern selection LP
  end if
end while
```

Observe that all patterns of size 1 and size 2 are part of the partial selection LP and that when generating patterns of size 3 and size 4 we always evaluate their reduced cost. Observe too that we do not only add patterns with a negative reduced cost when we generate patterns of size 3 and size 4. As one of our primary goals is to limit the number of pattern generation passes, we do not want to be too conservative. Ideally, the values ρ_3 and ρ_4 are set based on an analysis of the instance that needs to be solved. However, we have been unable to develop methodology to do so and have take a more pragmatic approach. First, because we have access to a machine with 16Gb of memory the number of patterns of size 3 does not pose a problem and we have used $\rho_3 = \infty$, which means we add all patterns of size 3. Second, after experimenting with a few instances and a few values for ρ_4, we found that $\rho_4 = 20$ performed well. In fact, with $\rho_4 = 20$, all instances were solved in two pattern

generation passes, i.e., one pass to generate the partial instance, and one pass to verify optimality of the solution produced. The effect of only incorporating patterns of size 4 with a reduced cost of less than or equal to 20 ($= \rho_4$) is dramatic as can be seen in Table 3. The number of patterns generated is reduced by a factor of more than 20.

Table 3: Number of patterns

Instance	# of 1-stop patterns	# of 2-stop patterns	# of 3-stop patterns	# of 4-stop patterns	# of 4-stop patterns with r.c. $\leq \rho_4$
1	136	7,163	220,949	2,801,732	32,213
2	106	5,377	154,719	3,235,373	82,964
3	157	8,225	279,217	4,048,867	75,466
4	129	7,464	284,176	5,089,771	240,997
5	169	9,015	312,962	5,098,761	44,265
6	147	8,414	331,629	8,497,947	36,152
7	157	10,281	422,653	8,542,524	114,402
8	194	16,356	784,857	15,838,715	276,416

CHAPTER III

COMPUTATIONAL EXPERIMENTS

The research reported in this thesis was motivated by our long-time collaboration with Praxair a producer and distributor of industrial gases. Praxair used the simple bound discussed in Section 2.2 to get an idea of the performance of their distribution strategy and to get a sense of the potential savings if additional resources and efforts were invested in improving their distribution strategy. As mentioned, this simple bound has two main deficiencies: it ignores the different storage capacities at customers and it assumes that deliveries can be perfectly timed. Our work eliminates the first deficiency.

We conducted various computational experiments to analyze the effect on the lower bound on the minimum total mileage required to satisfy demand of explicitly taking varying storage capacities into account. The data used in our experiments had usage information for about 2000 customers served from 36 plants (with the smallest plant serving about 10 customers and the largest plant serving about 150 customers). Each customer is supplied from one particular plant. Consequently, we are dealing with independent 36 instances.

The primary experiment involved computing increasingly tighter lower and upper bounds on D^* the bound on the minimum total mileage required to satisfy demand. The results are displayed in Figure 8.

First, the results shows that limiting ourselves to patterns with at most three or four customers is sufficient to obtain tight bounds on D^*. Second, the results show that allowing more deliveries per trip has a substantial effect on the upper bound, but hardly any effect on the lower bound. The latter result was somewhat counter to our expectations, but has important implications because it suggests that investing in larger storage facilities at customers, which is often discussed as a potential way of reducing distribution costs, may not deliver the desired savings. Finally, by comparing the actual incurred mileage to LB_4 over a period of time, Praxair will gain insight in the effectiveness of its distribution strategy

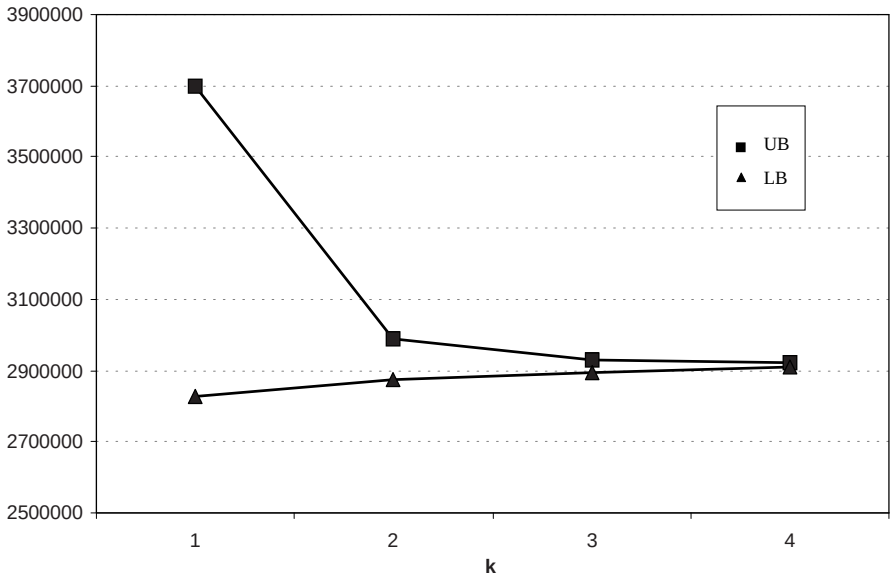


Figure 8: Lower and upper bounds on D^*

and in the potential savings that may result from improvements to its distribution strategy.

Next, we investigate whether the behavior observed for the complete system is also observed at the individual plant level. Table 6 shows the relative gap between the lower and upper bounds obtained for $k = 3$ and $k = 4$, i.e., $\frac{UB_3 - LB_3}{UB_3} \times 100$ and $\frac{UB_4 - LB_4}{UB_4} \times 100$, for each plant.

We see that for bounds LB_4 and UB_4 the largest relative gap is 2.53% for Plant 18 and the smallest relative gap is 0.02% from Plant 1. To understand the cause of the differences, we examined these plants in more detail. Two factors clearly impact the difference between the value of LB_4 and UB_4:

- The number of customers with $C_i < \frac{Q}{4}$
- The number of times we have to make deliveries to customers with $C_i < \frac{Q}{4}$

Note that when all customers served by a plant have $C_i \geq \frac{Q}{4}$, then we have $LB_4 = UB_4$. When we look more closely at Plant 1, we see that when a direct delivery policy would be employed, the number of deliveries is 568.4 (computed as $\sum_i \frac{u_i}{min(C_i, Q)}$). Among these

Table 4: Relative gap between lower and upper bounds for $k = 3, 4$

Plant	Relative gap for $k = 3$	Relative gap for $k = 4$
1	0.24%	0.02%
2	0.13%	0.03%
3	2.25%	1.32%
4	0.87%	0.27%
5	3.89%	1.33%
6	1.84%	0.42%
7	1.51%	0.47%
8	2.10%	0.94%
9	0.60%	0.10%
10	1.58%	0.77%
11	0.80%	0.37%
12	0.97%	0.48%
13	0.95%	0.54%
14	2.67%	1.17%
15	0.15%	0.06%
16	0.63%	0.19%
17	0.98%	0.22%
18	10.05%	2.53%
19	0.50%	0.27%
20	0.14%	0.11%
21	1.09%	0.23%
22	0.24%	0.13%
23	2.86%	1.35%
24	1.03%	0.40%
25	0.79%	0.54%
26	0.86%	0.53%
27	3.22%	1.43%
28	1.08%	0.55%
29	2.38%	0.73%
30	0.94%	0.27%
31	0.93%	0.24%
32	1.53%	0.66%
33	0.35%	0.14%
34	0.08%	0.04%
35	1.30%	0.54%
36	0.36%	0.17%
Overall	1.20%	0.50%

554.0 correspond to deliveries to customers with $C_i \geq \frac{Q}{4}$, i.e., 97.5% of the total number if deliveries. On the other hand, for Plant 18 the number of deliveries is 163.8 when a direct delivery policy is employed, out of which 100.9 correspond to deliveries to customers with

$C_i \geq \frac{Q}{4}$, i.e., only 61.6% of the total number of deliveries.

Finally, we examine the improvements in the lower bounds LB_k at the individual plant level. Table 5 shows the percentage increase between LB_k and LB_{k+1} for $k = 1, 2, 3$ ($\frac{LB_{k+1}-LB_k}{LB_k} \times 100$) and the overall percentage increase between LB_1 and LB_4 ($\frac{LB_4-LB_1}{LB_1} \times 100$) for each plant.

The largest percentage increase is 27.73% for Plant 2 and the smallest percentage increase is 0.61% for Plant 34. Again, we can explain this difference by analyzing what happens when a direct delivery policy is employed. When a direct delivery policy is employed, Plant 2 has to make 52.8 deliveries. Among these 46.8 are to customers with $C_i \geq \frac{Q}{4}$ (with 2.7 to customers with $C_i \geq Q$), i.e., 88.6% (5.1%) of the total number of deliveries. On the other hand, Plant 34 has to make 420.4 when a direct delivery policy is employed, out of which 409.6 deliveries are to customers with $C_i \geq \frac{Q}{4}$ (with 350.0 to customers with $C_i \geq Q$), i.e., 97.4 % (83.3%) of the total number of deliveries.

The system wide percentage increase is only 2.89%. There are two reasons why the increase in the value of the lower bound is relatively small. First, 73.1% of the volume has to be delivered to customers with $C_i \geq Q$. Second, the geography of customers with $C_i < Q$ is such that they can be combined into delivery trips that do not increase the total mileage by much (the pattern selection LP will identify optimal combinations of customers).

Table 5: Lower bound increases

Plant	Percentage increase for $k = 1$	Percentage increase for $k = 2$	Percentage increase for $k = 3$	Percentage increase overall
1	2.06%	0.49%	0.22%	2.79%
2	21.64%	4.90%	0.10%	27.73%
3	3.91%	0.76%	0.58%	5.31%
4	2.03%	0.93%	0.48%	3.48%
5	5.20%	1.49%	1.28%	8.14%
6	2.83%	1.04%	0.34%	4.26%
7	1.82%	0.76%	0.71%	3.33%
8	1.79%	0.77%	0.88%	3.47%
9	1.38%	0.35%	0.34%	2.08%
10	1.61%	0.77%	0.67%	3.07%
11	1.52%	0.59%	0.38%	2.51%
12	4.34%	2.26%	0.39%	7.12%
13	0.66%	0.42%	0.32%	1.41%
14	3.52%	1.37%	0.89%	5.87%
15	0.71%	0.24%	0.06%	1.01%
16	1.84%	0.76%	0.35%	2.97%
17	0.86%	0.76%	0.76%	2.41%
18	5.70%	4.81%	5.67%	17.07%
19	0.41%	0.27%	0.18%	0.87%
20	1.12%	0.19%	0.03%	1.34%
21	4.54%	1.39%	0.81%	6.84%
22	1.34%	0.14%	0.09%	1.57%
23	18.31%	6.90%	0.84%	27.53%
24	2.13%	0.61%	0.49%	3.25%
25	1.26%	0.33%	0.25%	1.86%
26	0.96%	0.34%	0.30%	1.61%
27	1.83%	1.24%	0.97%	4.09%
28	3.94%	1.69%	0.21%	5.92%
29	2.88%	1.13%	0.49%	4.55%
30	1.25%	0.60%	0.34%	2.21%
31	1.39%	0.61%	0.38%	2.40%
32	0.84%	0.65%	0.37%	1.87%
33	0.17%	0.25%	0.21%	0.63%
34	0.49%	0.08%	0.04%	0.61%
35	2.81%	0.65%	0.77%	4.28%
36	1.60%	0.43%	0.20%	2.24%
overall	1.72%	0.71%	0.44%	2.89%

CHAPTER IV

OTHER USES OF THE TECHNOLOGY

In addition to providing a lower bound on the total mileage required to satisfy demand, the technology may have other benefits. For example, the selected base patterns may suggest effective practical delivery trips.

We demonstrate this potential benefit by examining a small instance introduced by Fisher et al. [21, 9] to illustrate the complexity of inventory routing problems.

Example. Consider the instance depicted Figure 9.

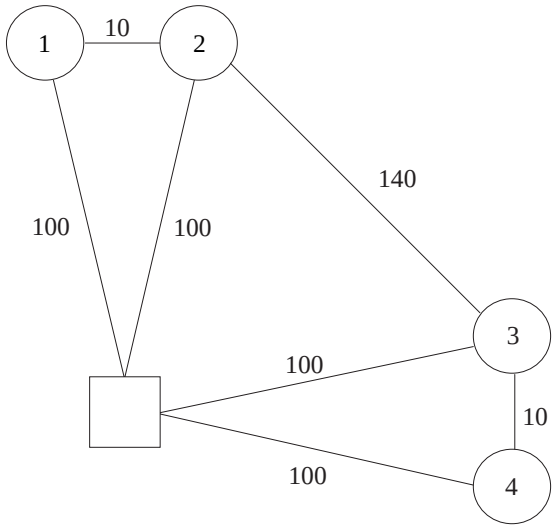


Figure 9: An example with 4 customers.

The vehicle capacity is 5000 and customer tank capacity and usage data is as follows:

Customer	C_i	u_i
1	5000	1000
2	3000	3000
3	2000	2000
4	4000	1500

The relevant optimal tour costs can be derived from the network shown, e.g., the optimal tour costs for visiting customers 1 and 2, denoted by c_{12}, is equal to \$210. A simple schedule jointly replenishes customers 1 and 2 as well as customers 3 and 4 on a daily basis. This schedule is natural because 1 and 2 (3 and 4, respectively) are near each other. Each customer i receives a quantity equal to its daily consumption u_i. The long-run average cost of this schedule is 420 miles per day. An improved schedule consists of a cycle that repeats every two days. On the first day, one trip is taken that replenishes 3000 gallons to 2 and 2000 gallons to 3, at a cost of 340 miles. On the second day, two trips are taken. The first trip replenishes 2000 gallons to 1 and 3000 gallons to 2. The second trip replenishes 2000 gallons to 3 and 3000 gallons to 4. Each trip costs 210 miles. The long-run time average cost of this schedule is 380 miles per day, which is nearly 10% lower than the first schedule. Fisher et al. observe that though it is easy to verify that the second schedule is better than the first, it is not at all obvious how to derive the second schedule.

By assuming a time period of a single day, the following pattern selection LP will be constructed, where we have left out routes visiting customer 1 and 3, 2 and 4, and 1 and 4, as no distance information is provided that allows the calculation of the tour length:

$$
\begin{array}{rrrrrrrrrl}
min & 200x_1 & +200x_2 & +200x_3 & +200x_4 & +210x_5 & +340x_6 & +210x_7 & +210x_8 & \\
s.t. & 5000x_1 & & & & +2000x_5 & & & & \geq 1000 \\
& & 3000x_2 & & & +3000x_5 & +3000x_6 & & & \geq 3000 \\
& & & 2000x_3 & & & +2000x_6 & +2000x_7 & +1000x_8 & \geq 2000 \\
& & & & 4000x_4 & & & +3000x_7 & +4000x_8 & \geq 1500 \\
& x \geq 0 & & & & & & & &
\end{array}
$$

The optimal solution selects patterns 5, 6, and 7 with value 0.5, which, given a time period of a single day, can be interpreted as using these patterns every other day. This corresponds precisely to the improved solution presented by Fisher et al. Our approach

also shows that no better solution exists!

The above example illustrates that the technology can be used for purposes other than performance measurement. The solution may suggest routing patterns that have not been considered so far. The technology may also be used to assist in tactical and strategic decisions. For example, it may be used to evaluate capital investment decisions related to increasing storage capacity at customers, or it may be used to evaluate customer - plant alignments in situations with multiple production facilities servicing the set of customers.

PART II

Inventory Routing with Continuous Moves

CHAPTER V

INVENTORY ROUTING WITH CONTINUOUS MOVES

5.1 Introduction

Most of the inventory routing literature deals with the repeated distribution of a single product from a single production facility to a set of customers with a fleet of homogeneous vehicles over a certain planning horizon. Even in situations with multiple production facilities serving the set of customers, the customers are usually aligned with a particular production facility resulting in the need to solve several single facility inventory routing problems. It is implicitly assumed that the production capacity at each facility is sufficient to serve its aligned customers and that all customers can be visited on an out-and-back trip from the production facility with which it is aligned. Consequently, the inventory routing problem seeks to find tours that can be performed by a single driver on a particular day.

Unfortunately, this is not always the reality. For example, in the liquid gas industry, the production and distribution of Argon cannot be captured with these models. Argon is typically produced in only a few facilities and, as Argon is a co-product of Oxygen production, the rate at which Argon is produced is determined by the demand for Oxygen rather than the demand for Argon, which leads to mismatches between production availability and product demand. Furthermore, due to the fact that only few facilities produce Argon, many customer cannot be served using out-and-back tours from a facility. Consequently, a substantial portion of Argon customers is served using sleeper teams that are on the road for several days in a row covering huge geographic areas, picking up product at different facilities along the way, and moving almost continuously.

In this part of thesis, we introduce the inventory routing problem with continuous moves (IRP-CM), which is specifically designed to handle these additional complexities, i.e., limited product availabilities at facilities, customers that cannot be served using out-and-back

tours, and delivery tours that can last for several days. We develop an innovative randomized greedy algorithm for its solution, including linear programming based postprocessing technology, and demonstrate its effectiveness in an extensive computational study.

The remainder of Part II is organized as follows. In Section 5.2, we formally introduce the inventory routing problem with continuous moves. In Chapter 6, we gradually develop the randomized greedy algorithm heuristic we propose for its solution. In Chapter 7, we discuss the delivery volume optimization model used as a postprocessing step in the heuristic. In Chapter 8, we present a variety of computational experiments of the heuristic and the delivery volume optimization conducted with data sets derived from real-life Argon production and distribution information. In Chapter 9, we propose a time-discretized model for the IRP-CM to solve to optimality. In Chapter 10, we discuss various solution techniques including problem size reductions and specialized IP techniques for the time-discretized model. In Chapter 11, we present a variety of computational experiments conducted to see the effectiveness of solution technologies. We also present some experiments conducted to compare solutions of the heuristic with the solutions of the time-discretized model. Finally, in Chapter 12 we present an integrated approach using heuristics and optimization algorithms providing effective and efficient technology for solving inventory problems with continuous moves.

5.2 Problem Description

Consider the following variant of the inventory routing problem. A single product must be distributed from a set $\mathcal{P}$ of plants to a set $\mathcal{C}$ of customers. A set $\mathcal{V}$ of vehicles is available for the distribution. Each customer $i \in \mathcal{C}$ has a usage rate u_i, a local storage capacity C_i, and an initial inventory I_i^0 at time 0. Each plant $j \in \mathcal{P}$ has a production rate p_j, unlimited storage capacity, and an initial inventory I_j^0 at time 0. Each vehicle $k \in \mathcal{V}$ has a tank capacity of Q and becomes available at a location $i \in \mathcal{P} \bigcup \mathcal{C}$ (either at a plant or at a customer) at time t_k^0 with an initial volume v_k^0 of product in its tank. Travel time between locations $u \in \mathcal{P} \bigcup \mathcal{C}$ and $v \in \mathcal{P} \bigcup \mathcal{C}$ is t_{uv}, and the cost for traveling between u and v is c_{uv}. The objective is to minimize transportation costs over the planning horizon T while trying

to ensure that none of the customers experience a stockout.

CHAPTER VI

HEURISTICS FOR THE IRP-CM

In this chapter, we discuss the development of a randomized greedy heuristic for the inventory routing problem with continuous moves. Although the main ideas and concepts used are similar to those of popular insertion heuristics for vehicle routing and scheduling problems, their actual implementation is more complicated due to the complex nature of the inventory routing problem with continuous moves. At every iteration of the heuristic two basic questions are answered: which customer visit to schedule next and how to incorporate this customer visit in the current partial solution? Due to the relatively long planning horizon, we may need to schedule multiple customer visits as a customer may be visited several times during the planning horizon. Deciding how to incorporate a customer visit in the current partial solution can be quite difficult because it involves choosing a vehicle, choosing a delivery time (as the delivery time impacts the volume that can be delivered), and determining whether or not the chosen vehicle should visit a plant to pickup additional product before going to the customer's site (and, if so, from which plant to pickup product).

It is common in practice to enforce a minimum delivery quantity at a customer. Given that one of the primary key performance indicators is volume delivered per mile, enforcing a minimum delivery quantity makes sense. Especially for customers far away from a plant, it is intuitively clear that delivering only a small amount is undesirable as it implies you will have to go out for another visit soon. Of course, it is not as simple as that, because when two far-away customers are close together it may be wise to visit them both on one and the same trip potentially delivering a relatively small quantity to one of them. Therefore, any minimum delivery quantity enforced at a customer should be determined based on the customer's usage rate, storage capacity, distance from a plant and the vicinity of other customers. We have adopted the use of minimum delivery quantities as part of our greedy heuristic. However, this restriction is removed during a postprocessing step when we solve

a delivery volume optimization linear program that maximizes the total volume of product delivered for a given set of delivery routes.

6.1 Basic Greedy Heuristic

The primary concern of a supplier is to ensure product availability at its customers (the basis for every vendor managed inventory arrangement). This concern is reflected in the greedy heuristic. The customer to visit next is selected based on its *urgency*, which is defined as the time remaining before the customer will experience a stockout. The less time remains, the more urgent it is to schedule a visit to the customer. The selection of the vehicle to serve the customer is based on the size of the stockout, if any, and the increase in transportation costs resulting from the selection. If there are vehicles that can serve the customer without causing a stockout, then we select the one among those with the smallest increase in transportation costs. If none of the vehicles can serve the customer without causing a stockout, then we select the vehicle that causes the smallest stockout, i.e., the one that can get to the customer as early as possible.

Let $\underline{d_i}$ denote the minimum delivery quantity for customer i ($\underline{d_i} \leq \min\{Q, C_i\}$). During the execution of the heuristic, we keep for each customer i the following information: the time of the last delivery t_i^l, the inventory after the last delivery I_i^l, the next time at which the minimum delivery quantity can be delivered t_i^m, and the time at which a stockout will occur if no other deliveries are made $t_i^s = t_i^l + \frac{I_i^l}{u_i}$.

At the heart of the heuristic is the logic that determines what happens when a vehicle is selected to serve a customer. So assume that vehicle k is selected to serve customer i. We consider two situations. First, assume that the amount of product in the vehicle v_k is greater than or equal to $\underline{d_i}$. Let $t_k^a(i)$ be the earliest time that vehicle k can arrive at customer i. In that case, the time of delivery and the amount to be delivered are set as

- If $t_k^a(i) \leq t_i^m$, then deliver $\underline{d_i}$ at time t_i^m.
- If $t_i^m < t_k^a(i) \leq t_i^s$, then deliver $\min\{v_k, C_i - (I_i^l - (t_k^a(i) - t_i^l)u_i)\}$ at time $t_k^a(i)$.
- If $t_k^a(i) > t_i^s$, then deliver $\min\{v_k, C_i\}$ at time $t_k^a(i)$.

Next, suppose that the amount of product in the vehicle v_k is less than $\underline{d_i}$. In order to satisfy the minimum delivery requirement at customer i, vehicle k needs to visit a plant first. Let's assume the vehicle visits plant j and let $t_k^a(j)$ denote the earliest time that vehicle k can arrive at plant j. Furthermore, let t_j^l and I_j^l denote the time of the last pickup at plant j and the inventory after the last pickup at plant j. We assume that whenever a vehicle visits a plant, the vehicle's tank is filled up. Since the amount of product that has to be picked up is $Q - v_k$, we only need to decide the pickup time $t_k^p(j)$. First, suppose that $t_k^a(j) \geq t_j^l$, then $t_k^p(j) = \max\{t_k^a(j), t_j^l + \frac{(Q-v_k)-I_j^l}{p_j}\}$. The second term of the maximum is the first time that enough product is available at plant j to fill up vehicle k. When $t_k^a(j) < t_j^l$ the situation is more complicated. Note that this situation may occur, because we may have scheduled one or more pickups for other vehicles occurring after the earliest possible arrival time of vehicle k at plant j. Since we do not want to change the pickup times of the already scheduled pickups at plant j, we need to be careful when deciding on the pickup time $t_k^p(j)$. Let $t_j^1, t_j^2, ..., t_j^l$ denote the pickup times of already scheduled pickups at plant j after $t_k^a(j)$, let $p_j^1, p_j^2, ..., p_j^l$ denote the associated pickup quantities, and let $I_j^1, I_j^2, ..., I_j^l$ denote the associated inventories after pickup. Let $s^* = \min\{s | I_j^s, I_j^{s+1}, ..., I_j^l \geq Q - v_k\}$. If we schedule the new pickup between $t_j^{s^*-1}$ and $t_j^{s^*}$, then the pickup times of the already scheduled pickups at plant j remain valid. To be more precise, we schedule the new pickup at $t_k^p(j) = \max\{t_k^a(i), t_j^{s^*} - \frac{I_j^{s^*} + p_j^{s^*} - (Q-v_k)}{p_j}\}$, where the second term of the maximum represents the first point in time when enough product is available to fill up vehicle k (If $I_j^l < Q - v_k$, then $t_k^p(j) = t_j^l + \frac{(Q-v_k)-I_j^l}{p_j}$).

Vehicle k goes to customer i after filling up its tank, which implies an earliest possible arrival time $t_k^a(i)$ at customer i. The same logic as before can now be applied to determine the time of delivery and the amount to deliver. The resulting basic greedy heuristic is presented in Algorithm 1.

6.2 Enhanced Greedy Heuristic

In the description of the basic greedy heuristic, we have implicitly assumed that the delivery to customer i is scheduled at the end of the current route of vehicle k (see Figure 10).

Algorithm 2 Greedy Heuristic for IRP-CM

Initialize
while there are customers left that will experience a stockout before the end of the planning horizon **do**
  Select the most urgent customer i, i.e., the customer that would first experience a stockout
  Select the most appropriate vehicle k to visit customer i, i.e., if a stockout is unavoidable the vehicle that results in the shortest stockout, if a stockout can be avoided the vehicle that results in the smallest increase in transportation costs (a vehicle may have to visit a plant j to pickup additional product on its way to customer i)
  Update information to reflect the newly scheduled delivery (customer, vehicle, plant)
end while

Next, we describe an enhanced greedy heuristic in which we evaluate additional options for including the delivery to customer i in the route and schedule of vehicle k. Instead of only examining scheduling the delivery to customer i at the end of the current route of vehicle k, we examine scheduling the delivery to customer i at any point in the route *after* the last pickup. In doing so, we only allow another pickup when the delivery is scheduled at the end of the current route (see Figure 11).

Note that care has to be taken when evaluating insertions into the route other than at

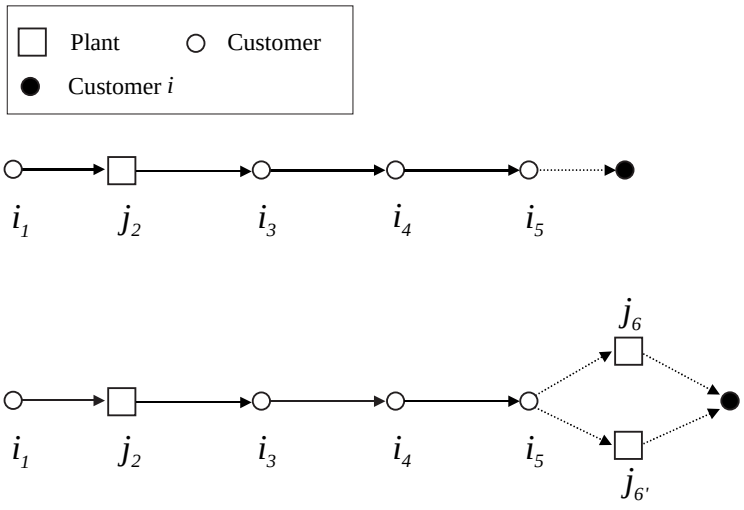


Figure 10: Simple route extension.

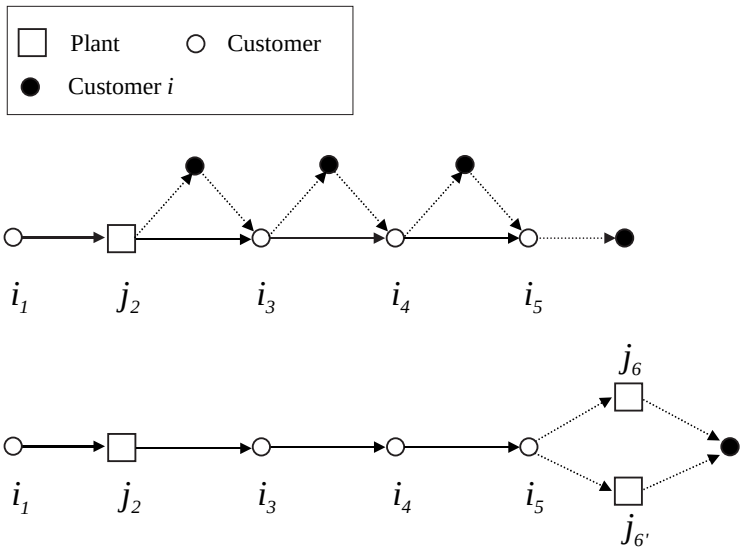


Figure 11: Simple route insertion.

the end to ensure that we do not incur stockouts at customers that will follow the delivery to customer i as the arrival times at their sites will change and that we can still deliver the minimum delivery quantity at these customers.

On top of considering insertions in addition to simple extension of a route, we introduce one more enhancement. Before scheduling a delivery to customer i by vehicle k, we remove the last currently scheduled delivery for vehicle k. Once the delivery of customer i has been scheduled, we proceed by scheduling the delivery of the the removed delivery, where we not only consider vehicle k, but all vehicles (see Figure 12).

6.3 Randomized Enhanced Greedy Heuristic

As the inventory routing problem with continuous moves is a highly complex problem, it is unlikely that a greedy heuristic will consistently find high quality solutions. A simple, yet powerful way to improve the performance of a greedy heuristic and at the same time increase its robustness is to introduce randomization into the heuristic. We have converted the enhanced greedy heuristic presented in the previous section into a greedy randomized adaptive search procedure (GRASP). For a survey of GRASP, see Resende [32]. The main

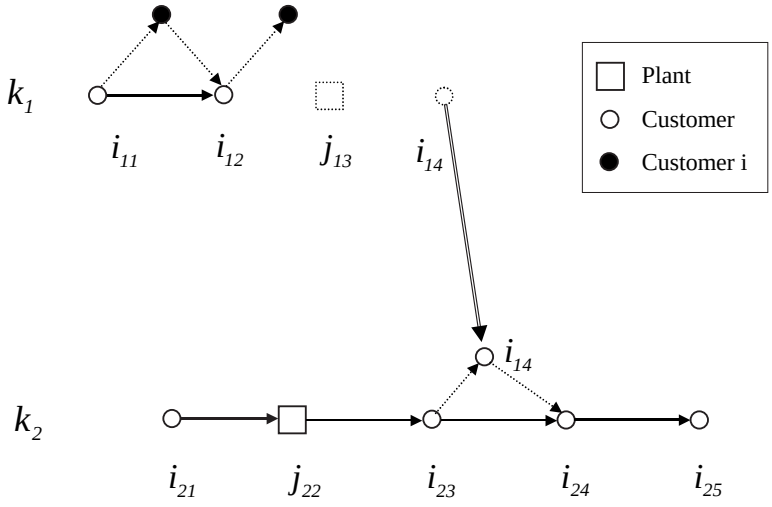


Figure 12: Advanced route insertion.

idea of a GRASP is to randomly select from among the best k choices (for some small value of k) instead of always choosing the best. The intuition is that some times choosing the second or even third best option may allow for much better future choices. A GRASP is run multiple times typically producing several different solutions. We adopt the best among the solutions produced. We have converted the enhanced greedy heuristic into a GRASP by randomly selecting among the k most urgent customers instead of always choosing the most urgent customer as the customer for which we schedule the next delivery. A high level over view can be found in Algorithm 2 (where RCL denotes Restricted Candidate List).

Here, N is the number of GRASP iterations. We have experimented with three different randomized selection schemes. The first gives equal probability to each customer in the RCL, i.e., each customer is selected with probability $\frac{1}{|RCL|}$. The second is biased towards the more urgent customers, i.e., the k^{th} most urgent customer is selected with probability $\frac{|RCL|-k+1}{0.5(|RCL|+1)|RCL|}$. The last is also biased towards the more urgent customers, but considers the customers' stockout times. If t_i^s denotes the stockout time of customer i, then the probability that customer i is selected is equal to $\frac{1}{|RCL|-1}(1-\frac{t_i^s}{\sum_j t_j^s})$.

Algorithm 3 A GRASP for IRP-CM

```
min_stockout = ∞
min_transportation = ∞
for iter = 1, 2, ..., N do
  Initialize
  while RCL is not empty do
    Randomly select customer i from RCL
    Schedule a delivery for customer i
    Update RCL
  end while
  Calculate stockout and transportation cost (SC and TC) of the schedule produced
  if SC < min_stockout then
    min_stockout = SC
    min_transportation = TC
  else
    if SC == min_stockout and TC < min_transportation then
      min_transportation = TC
    end if
  end if
end for
```

CHAPTER VII

DELIVERY VOLUME OPTIMIZATION

7.1 Introduction

The randomized greedy heuristic described above produces a complete delivery schedule for the planning period, i.e., for each vehicle a route specifying a sequence of visits to plants and customers and for each visit the time of the visit and the amount of product to pickup (in case of a plant visit) or to deliver (in case of a customer visit). The times of the visits and the amounts of product picked up or delivered are strongly influenced by the minimum delivery requirement at customers and the rigid logic for deciding the time of a visit. In this chapter, we develop a delivery volume optimization model (DVO), a linear program, that takes a global look at the schedule produced and maximizes the total volume of product delivered over the planning period by removing the minimum delivery requirement and reevaluating the times of visits and the amounts of product picked up or delivered. The model will not alter the vehicle routes and thus the total distance traveled remains unchanged. Therefore, the model results in an increase in the *volume delivered per mile*, the most popular performance measure for inventory routing problems. (Delivery volume optimization has been analyzed in the context of the traditional inventory routing problem by Campbell and Savelsbergh [13].)

7.2 Linear Programming Model for Delivery Volume Optimization

The delivery volume optimization model is based on the observation that the delivery schedule produced by the heuristic induces two sequences: for each vehicle a sequence of site visits and for each site a sequence of vehicle visits (where a site is either a plant or a customer) and that each visit appears exactly once in a site visit sequence and once in a vehicle visit sequence.

Let S_k denote the sequence of sites visited by vehicle k and let S_i denote the sequence of vehicles that visit site i, where i is either a plant of a customer. Let $S_k(j)$ represent j^{th} element of sequence S_k, where $S_k(0)$ denotes the position of vehicle k at time 0. Let $S_i(j)$ represent the j^{th} element of sequence S_i, where we define $t_{S_i(0)} = 0$. Furthermore, let N denote the set of all visits. Recall that each visit appears in one of the sequences S_k and also in one of the sequences S_i.

The delivery volume optimization model has the following decision variables:

t_n : time of visit n, $\forall n \in N$.

d_n : pickup or delivery quantity during visit n, $\forall n \in N$.

v_n : product volume of vehicle k right after visit n, $\forall k \in \mathcal{V}$ and $n \in S_k$.

$I_i^{t_n}$: inventory level of site i right after visit n, $\forall i \in \mathcal{P} \bigcup \mathcal{C}$ and $n \in S_i$.

I_i^T : inventory level of site i at time T. $\forall i \in \mathcal{P} \bigcup \mathcal{C}$.

Below the linear program representing the delivery volume optimization model.

$$z = \max \sum_{i \in \mathcal{C}, j=1,...,|S_i|} d_{S_i(j)} \tag{1}$$

$$s.t. \quad t_{S_k(j)} \geq t_{S_k(j-1)} + t_{S_k(j-1),S_k(j)}, \ \forall k \in \mathcal{V}, j = 1, 2, ..., |S_k|, \tag{2}$$

$$t_{S_i(j)} \geq t_{S_i(j-1)}, \ \forall i \in \mathcal{P} \bigcup \mathcal{C}, j = 1, 2, ..., |S_i|, \tag{3}$$

$$0 \leq t_n \leq T, \ \forall n \in N, \tag{4}$$

$$v_{S_k(j-1)} + d_{S_k(j)} = v_{S_k(j)}, \ \forall k \in \mathcal{V}, j = 1, ..., |S_k| : S_k(j) \in \mathcal{P}, \tag{5}$$

$$v_{S_k(j-1)} - d_{S_k(j)} = v_{S_k(j)}, \ \forall k \in \mathcal{V}, j = 1, ..., |S_k| : S_k(j) \in \mathcal{C}, \tag{6}$$

$$0 \leq d_n, \ \forall n \in N, \tag{7}$$

$$0 \leq v_{S_k(j)} \leq Q, \ \forall k \in \mathcal{V}, j = 1, ..., |S_k|, \tag{8}$$

$$I_i^{t_{S_i(j-1)}} - u_i(t_{S_i(j)} - t_{S_i(j-1)}) + d_{S_i(j)} = I_i^{t_{S_i(j)}}, \ \forall i \in \mathcal{C}, j = 1, ..., |S_i|, \tag{9}$$

$$I_i^{t_{S_i(|S_i|)}} - u_i(T - t_{S_i(|S_i|)}) = I_i^T, \ \forall i \in \mathcal{C}, \tag{10}$$

$$I_i^{t_{S_i(j-1)}} + p_i(t_{S_i(j)} - t_{S_i(j-1)}) - d_{S_i(j)} = I_i^{t_{S_i(j)}}, \ \forall i \in \mathcal{P}, j = 1, ..., |S_i|, \tag{11}$$

$$I_i^{t_{S_i(|S_i|)}} + p_i(T - t_{S_i(|S_i|)}) = I_i^T, \ \forall i \in \mathcal{P}, \tag{12}$$

$$0 \leq I_i^{t_{S_i(j-1)}} - u_i(t_{S_i(j)} - t_{S_i(j-1)}),\ \forall i \in \mathcal{C}, j = 1, 2, 3, ..., |S_i|, \tag{13}$$

$$0 \leq I_i^{t_{S_i(|S_i|)}} - u_i(T - t_{S_i(|S_i|)}),\ \forall i \in \mathcal{C}, \tag{14}$$

$$0 \leq I_i^{t_{S_i(j)}} \leq C_i,\ \forall i \in \mathcal{C}, j = 1, ..., |S_i|, \tag{15}$$

$$0 \leq I_i^{T} \leq C_i,\ \forall i \in \mathcal{C}, \tag{16}$$

$$0 \leq I_i^{t_{S_i(j)}}\ \forall i \in \mathcal{P}, j = 1, ..., |S_i|, \tag{17}$$

Constraints (2) ensure that time difference between two consecutive visits of a vehicle is greater than the time it takes to travel from one site to the other. Constraints (3) ensure the proper time precedence for visits at each site. Constraints (4) ensure that all scheduled deliveries and pickups take place before the end of the planning horizon. Constraints (5) - (8) track the product volume of the vehicles and enforce vehicle capacity. Constraints (9) - (10) track product inventory at customers and constraints (11) - (12) track product inventory at plants. Constraints (13) - (14) guarantee no stockouts at customers. The remaining constraints provide bounds on inventory at customers and plants. The objective (1) maximizes total volume of product delivered at customers.

If the heuristic has produced a schedule without stockouts, then the schedule after delivery volume optimization will have no stockouts either. However, when the schedule produced by the heuristic has stockouts, it is possible that after delivery volume optimization the schedule will have no stockouts. If the delivery optimization linear program is infeasible, it proves that no schedule without stockouts exists for these vehicle and site sequences.

CHAPTER VIII

COMPUTATIONAL RESULTS

We conducted various computational experiments to analyze the performance of the different greedy heuristics and to establish the value of the delivery volume optimization model. The base instance used in our experiments involved 7 production facilities, about 200 customers, and a homogeneous fleet of 7 vehicles. We have historical information about production rates at plants and usage rates at customers as well as vehicle capacities and storage capacities at customers. (The data was provided by Praxair Inc., a producer and distributor of industrial gases and long-time member of the Leaders in Logistics program at Georgia Tech.)

We derived two data sets from the base instance. In the first data set of 10 instances ($i1$, $i2$, ..., $i10$), we randomly generate the initial inventory levels for customers at the start of the planning period. In the second data set of 10 instances ($l1$, $l2$, ..., $l10$), we randomly generate locations for the customers. The planning horizon used in the experiments is 10 days, unless specifically stated otherwise. Furthermore, for all experiments involving a rolling horizon, we implement only the decisions for the first 5 days and then roll the horizon forward by 5 days.

8.1 Comparison of the Greedy Heuristics

The first set of experiments aims to determine the value of the different levels of sophistication introduced on top of the basic greedy heuristic. Therefore, we compare the performance of the basic greedy heuristic (BGH), the enhanced greedy heuristic (EGH), and the randomized enhanced greedy heuristic (RGH).

A small experiment was conducted first to determine appropriate settings for the randomized enhanced greedy heuristic. Even though the differences in performance for the various settings that we experimented with were not substantial, the best performance was

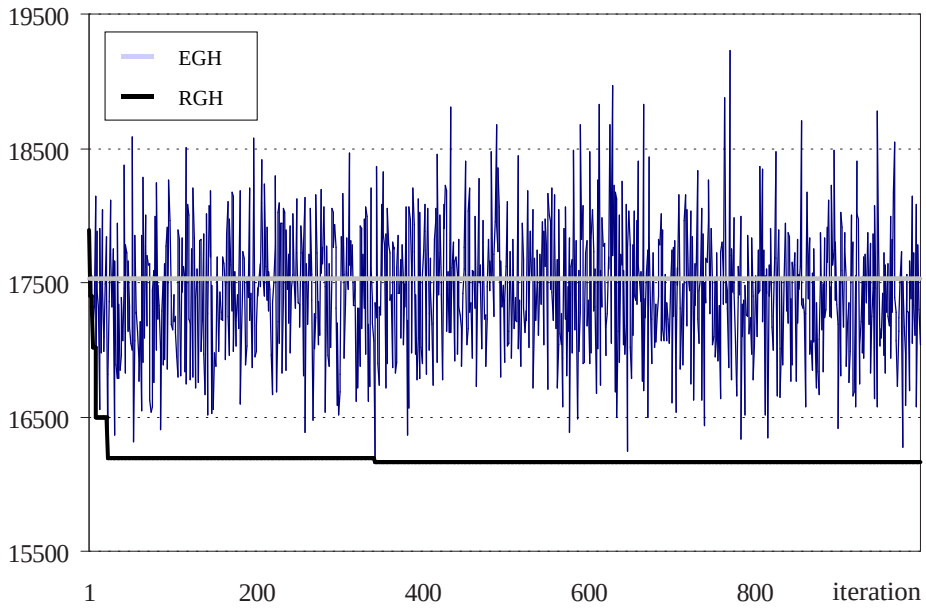


Figure 13: Typical Behavior of the Randomized Enhanced Greedy Heuristic

obtained with a restricted candidate list of size 4 and selecting among the candidates using the second scheme proposed, i.e., with a simple bias towards the more urgent customers. Consequently, these setting are used in all the experiments on which we will report. Figure 13 displays the typical behavior of the randomized enhanced greedy heuristic as it progresses over time. It shows the value of the solution obtained, i.e., the total transportation costs, for each iteration. We see that it is possible to obtain a high quality solution with a relatively small number of iterations (randomization improves the solution quality by about 7%). Therefore, we have chosen to limit the number of iterations of the randomized enhanced greedy heuristic to 200 in our experiments.

As an aside, we note that randomization may also help to achieve solutions without stockouts. For example, on one instance (not in one of the data sets), BGH constructed a solution with a stockout quantity of 490.0, EGH produced a solution with a stockout quantity of 20.1, and RGH found a solution with no stockout.

Figure 14 shows the transportation costs of the solutions produced by the different

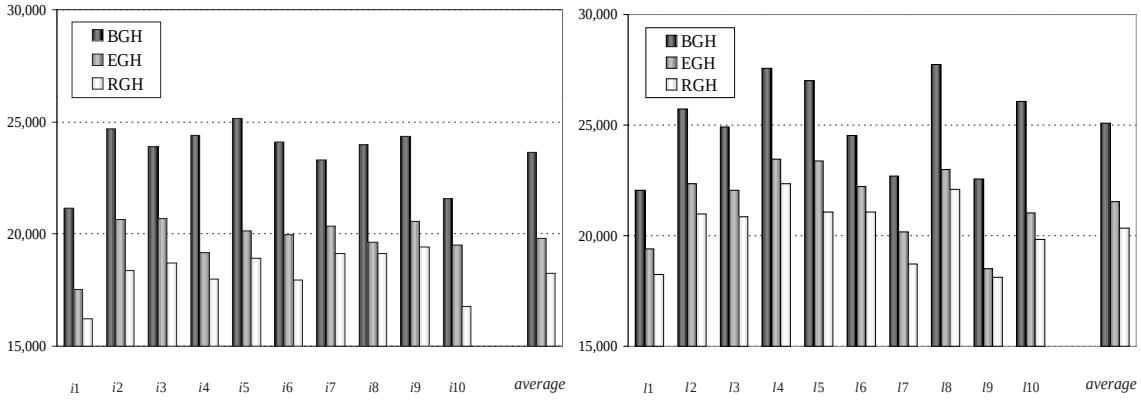


Figure 14: Transportation Costs for the Different Heuristics

heuristics for each of instances in our two data sets. More precisely, the first bar represents the transportation costs of the solution produced by the basic greedy heuristic, the second bar represents the transportation costs of the solution produced by the enhanced greedy heuristic, and the third bar represents the transportation costs of the solution produced by the randomized enhanced greedy heuristic. The details can be found in Table 6.

Table 6 shows the transportation costs (TC) of the solutions produced by the different heuristics for each of the instances in our two data sets. More precisely, the column headed TC_{BGH} gives the transportation costs of the solution produced by the basic greedy heuristic, the column headed TC_{EGH} gives the transportation costs of the solution produced by the enhanced greedy heuristic, and the column headed TC_{RGH} gives the transportation costs of the solution produced by the randomized enhanced greedy heuristic. The last two columns show the percentage decrease in transportation costs when switching from BGH to EGH and when switching from EGH to RGH, respectively (computed as $\frac{TC_{BGH}-TC_{EGH}}{TC_{BGH}} \times 100$, and $\frac{TC_{EGH}-TC_{RGH}}{TC_{EGH}} \times 100$ respectively).

The results in Figure 14 and Table 6 clearly demonstrate the benefits of a more flexible insertion mechanism (about a 15% improvement) and of randomization (about a 7% improvement).

Table 6: Transportation Costs for the Different Heuristics

Instance	TC_{BGH}	TC_{EGH}	TC_{RGH}	Percentage decrease from TC_{BGH} to TC_{EGH}	Percentage decrease from TC_{EGH} to TC_{RGH}
$i1$	21,148	17,532	16,216	17.10%	7.51%
$i2$	24,681	20,646	18,365	16.35%	11.05%
$i3$	23,870	20,678	18,695	13.37%	9.59%
$i4$	24,406	19,165	17,997	21.47%	6.09%
$i5$	25,161	20,155	18,926	19.89%	6.10%
$i6$	24,117	19,976	17,946	17.17%	10.17%
$i7$	23,292	20,358	19,125	12.60%	6.06%
$i8$	23,984	19,626	19,128	18.17%	2.54%
$i9$	24,351	20,567	19,422	15.54%	5.57%
$i10$	21,566	19,522	16,753	9.48%	14.18%
average	23,658	19,823	18,257	16.21%	7.90%
$l1$	22,049	19,412	18,253	11.96%	5.97%
$l2$	25,725	22,329	20,989	13.20%	6.00%
$l3$	24,923	22,055	20,849	11.51%	5.47%
$l4$	27,566	23,467	22,361	14.87%	4.71%
$l5$	27,023	23,364	21,052	13.54%	9.90%
$l6$	24,519	22,204	21,061	9.44%	5.15%
$l7$	22,702	20,183	18,710	11.10%	7.30%
$l8$	27,746	22,990	22,099	17.14%	3.87%
$l9$	22,585	18,490	18,125	18.13%	1.98%
$l10$	26,071	21,005	19,811	19.43%	5.69%
average	25,091	21,550	20,331	14.11%	5.66%

8.2 Value of Delivery Volume Optimization

The delivery volume optimization model maximizes the total volume of product delivered over the planning period by removing the minimum delivery requirement and reevaluating the times of visits and the amounts of product picked up or delivered.

To establish the value of delivery volume optimization, we took the solutions produced by the randomized enhanced greedy heuristic, extracted the sequences of site visits for the vehicles and the sequences of vehicle visits for the sites, and solved the delivery volume optimization linear program. The results are presented in Figure 15.

More precisely, the first bar represents the product quantity delivered in the solution produced by the randomized enhanced greedy heuristic and the second bar represents the product volume delivered after the delivery times and delivery quantities are optimized. The details can be found in Table 7.

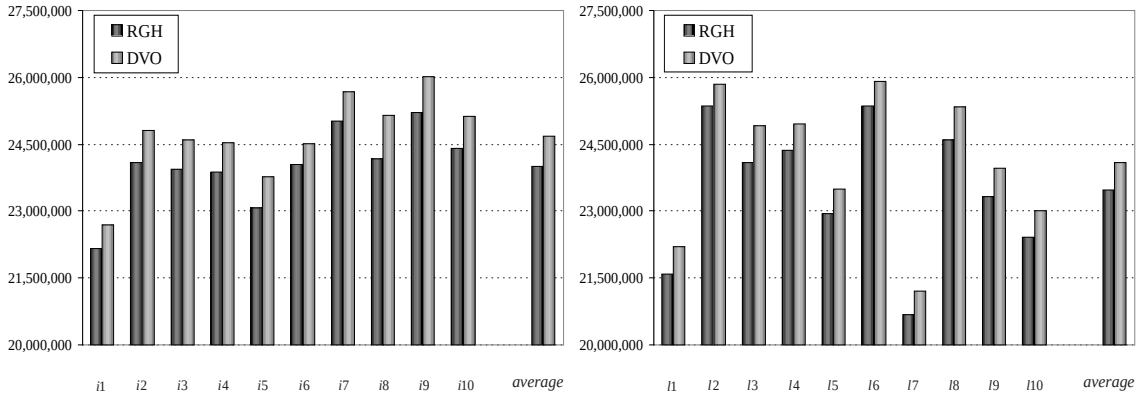


Figure 15: The Value of Delivery Volume Optimization

Table 7 shows the percentage increase between the product quantity delivered in the solution produced by the randomized enhanced greedy heuristic (PQ_{RGH}) and the product volume delivered after the delivery times and delivery quantities are optimized (PQ_{DVO}), computed as $\frac{PQ_{DVO}-PQ_{RGH}}{PQ_{RGH}} \times 100$.

Table 7: The Value of Delivery Volume Optimization

Instance	PQ_{RGH}	PQ_{DVO}	Percentage increase
$i1$	22,167,900	22,686,800	2.34%
$i2$	24,097,400	24,819,200	3.00%
$i3$	23,936,300	24,592,500	2.74%
$i4$	23,881,800	24,543,600	2.77%
$i5$	23,079,400	23,779,600	3.03%
$i6$	24,050,500	24,512,800	1.92%
$i7$	25,018,600	25,679,900	2.64%
$i8$	24,175,500	25,138,100	3.98%
$i9$	25,212,600	26,012,000	3.17%
$i10$	24,400,100	25,121,000	2.95%
average	24,002,010	24,688,550	2.86%
$l1$	21,590,200	22,204,500	2.85%
$l2$	25,353,600	25,840,900	1.92%
$l3$	24,091,300	24,914,300	3.42%
$l4$	24,374,300	24,962,900	2.41%
$l5$	22,942,000	23,485,200	2.37%
$l6$	25,355,800	25,915,400	2.21%
$l7$	20,667,400	21,207,600	2.61%
$l8$	24,596,100	25,331,600	2.99%
$l9$	23,329,400	23,956,600	2.69%
$l10$	22,409,300	23,005,200	2.66%
average	23,470,940	24,082,420	2.61%

The largest percentage increase is 3.98% for instance $i8$ and the smallest percentage increase is 1.92% for instance $i6$ and $l2$. As we mentioned before, the transportation costs for both solutions are identical. Therefore, delivery volume optimization results in a higher *volume delivered per mile.*

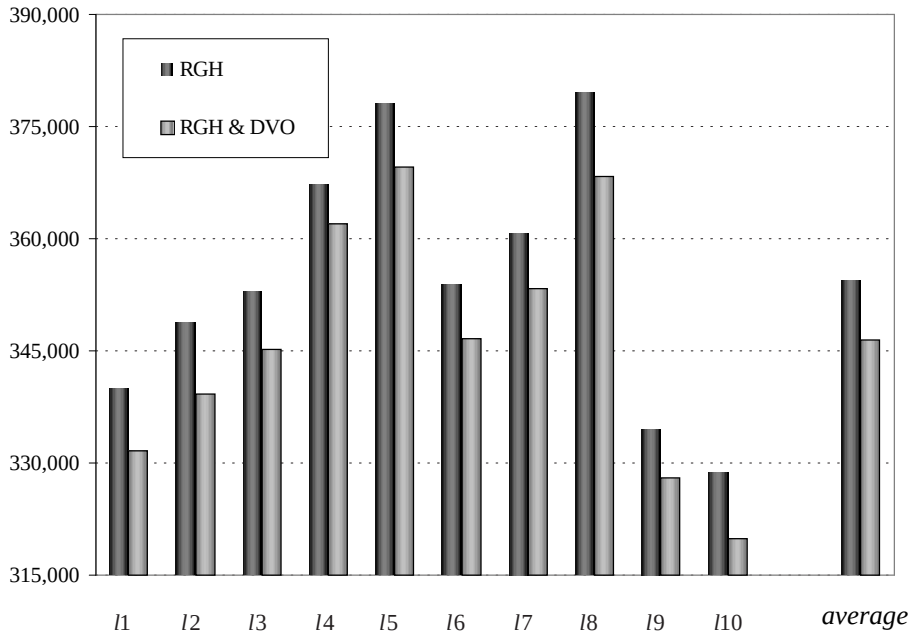


Figure 16: Long Term Effect of Delivery Volume Optimization

For a short planning period, maximizing the volume delivered may be meaningful, but for a long planning period it is not. For a long planning period, the volume that can be delivered depends almost entirely on customer usage, which is beyond our control (in a short planning period, we can effectively exploit the storage capacity at a customer). In practice, however, inventory routing problems are usually solved using a rolling horizon framework. Therefore, we have investigated the value of incorporating delivery volume optimization in a solution approach for a short-term planning problem that is embedded in a rolling horizon framework. More precisely, we solve a 10-day problem and implement the schedule for the first 5 days before rolling the planning horizon forward by 5 days. Using this rolling horizon framework to plan a period of 5 months, we simulate two different

setups. In the first setup, each 10-day problem is solved using the randomized enhanced greedy heuristic without delivery volume optimization. In the second setup, each 10-day problem is solved using the randomized enhanced greedy heuristic followed by delivery volume optimization. Figure 16 presents the results of this experiment. More precisely, the first bar represents the total transportation costs for the 5-month period in the solution produced by the randomized enhanced greedy heuristic when it is embedded in a rolling horizon framework, and the second bar represents the total transportation costs for the 5-month period in the solution produced by the randomized enhanced greedy heuristic when it is embedded in a rolling horizon framework and delivery volume optimization is applied to each intermediate solution generated. The details can be found in Table 8.

Table 8 shows the transportation costs of the solutions produced by the randomized enhanced greedy heuristic embedded in a rolling horizon framework with and without delivery volume optimization for each of the instances in our second data set. More precisely, the first column ($TC_{RGH\ RH}$) lists the total transportation costs for the 5-month period in the solution produced by the randomized enhanced greedy heuristic when it is embedded in a rolling horizon framework without delivery volume optimization. The second column ($TC_{RGH\&DVO\ RH}$) lists the total transportation costs for the 5-month period in the solution produced by the randomized enhanced greedy heuristic when it is embedded in a rolling horizon framework and delivery volume optimization is applied to each intermediate solution generated. The final column lists the percentage decrease.

We observe that the transportation costs over the 5-month period are consistently and substantially lower when incorporating delivery volume optimization (the differences in product volumes delivered are negligible, as expected). The reason for this, most likely, is that by incorporating delivery volume optimization, the short term planning problems can focus on how to best serve the customers that need to be served and do so in a cost-effective way (low transportation cost) without having to worry about less important customers (during that planning period). Consequently, the use of delivery volume optimization also results in a higher *volume delivered per mile* for long planning periods.

Table 8: Long Term Effect of Delivery Volume Optimization

Instance	$TC_{RGH\ RH}$	$TC_{RGH\&DVO\ RH}$	Percentage decrease
$l1$	339,957	331,569	2.47%
$l2$	348,708	339,162	2.74%
$l3$	352,932	345,129	2.21%
$l4$	367,317	361,951	1.46%
$l5$	378,116	369,602	2.25%
$l6$	353,890	346,604	2.06%
$l7$	360,734	353,308	2.06%
$l8$	379,535	368,325	2.95%
$l9$	334,601	328,045	1.96%
$l10$	328,744	319,909	2.69%
average	354,453	346,361	2.28%

8.3 Impact of Minimum Delivery Quantities

All variants of the greedy heuristic enforce a minimum delivery quantity at a customer. In the experiments presented up to now, this minimum delivery quantity was set to the "standard fill" for a customer. Standard fill is a delivery quantity based on historical data and used by planners as a target delivery amount. For our data sets, standard fill is approximately 90% of the maximum delivery quantity.

In order to investigate the effects of the minimum delivery quantity on the performance of the heuristics algorithms, either standard or embedded in a rolling horizon framework, we solved the base instance for different minimum delivery quantity settings. The minimum delivery quantities ranged from 60% to 96% of the maximum possible delivery quantity at a customer i, i.e., $\min\{Q, C_i\}$ (60%, 70%, 80%, 82%, 84%, 86%, 88%, 90%, 92%, 94%, and 96%). To be more precise, we used the enhanced greedy heuristic, the randomized enhanced greedy heuristic, the randomized enhanced greedy heuristic embedded in a rolling horizon framework, and the randomized enhanced greedy heuristic with delivery volume optimization embedded in a rolling horizon framework, to construct a delivery plan for the base instance for a 5-month period. (When a heuristic is embedded in a rolling horizon framework, we solve for 10 days, implement 5 days, and roll forward 5 days. When a heuristic is not embedded in a rolling horizon framework, we solve for the entire planning period of 5 months.) The results are presented in Figure 17.

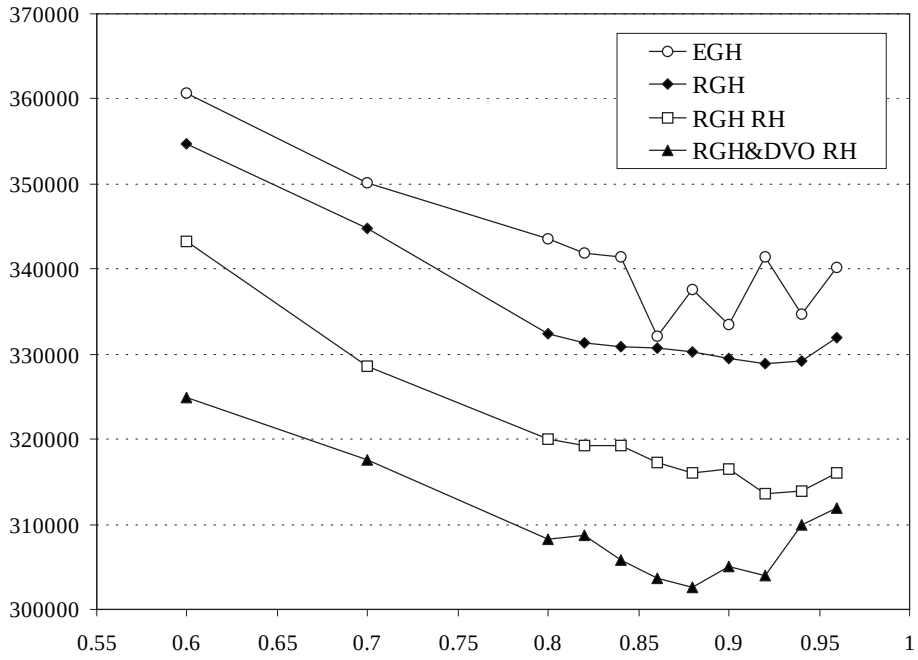


Figure 17: Impact of Minimum Delivery Quantities

The results show that a fairly high minimum delivery quantity, between 85 and 95 percent of the maximum delivery quantity, has to be used to obtain the best performance for the heuristics. This seems to be in line with the "standard fill" used as a guideline in practice, which is a round 90%. When the minimum delivery quantity is set to small, the heuristics schedule deliveries too early leading to too many small deliveries. When the minimum delivery quantity is set too large, we cannot take advantage of the advantages provided by vendor managed inventory resupply, because there is hardly any flexibility in the timing and delivery quantity.

We also observe that using the randomized enhanced greedy heuristic in a rolling horizon framework is better than using it to construct a solution for the entire 5-month planning period. This is likely caused by the fact that randomization had been given a greater chance. When embedded in a rolling horizon framework, 30 short term planning problems are solved and *each* of them is solved 200 times. Without the rolling horizon framework the long term planning problem is solved 200 times.

CHAPTER IX

A TIME-DISCRETIZED MODEL FOR THE IRP-CM

In this chapter, we propose a mixed integer programming model for the optimal solution of the IRP-CM.

Set partitioning and set covering approaches have been used with some success for solving traditional inventory routing problems, e.g., Fisher et al. [21] and Bell et al.[9]. Such approaches are based on generating all (or at least most) potential delivery tours and then selecting a minimum cost set of tours that ensures that customers will not run out of product. Doing so is much more complicated than in standard vehicle routing contexts as it also involves determining delivery quantities (and the maximum delivery quantity depends on the time of the delivery). The use of set partitioning and set covering approaches for the inventory routing problem with continuous moves is even more complex (and likely inappropriate), because the number of potential tours is even greater. This is because there is no limit on tour length (except for the length of the planning period) and customers can be visited more than once. Therefore, we have chosen to model the inventory routing problem with continuous moves using a time-indexed integer programming formulation similar to those used to formulate complex scheduling problems. This is not unreasonable as the inventory routing problem with continuous moves can be viewed as a problem of scheduling visits to plants and customers.

The model is based on constructing a network for each vehicle, in which nodes represent a visit to a site at a particular time. The use of time indices allows us to keep track of the inventory levels for customers as well as plants.

9.1 Time Discretization

We use t to denote a time index, and let $\{1, 2, \ldots, T\}$ be the set of time indices. If a time unit for the time discretization is 1 hour, then time index $t = 7$ indicates seven hours later

than the initial time.

We assume that t_{uv}, travel time between locations $u \in \mathcal{P} \bigcup \mathcal{C}$ and $v \in \mathcal{P} \bigcup \mathcal{C}$, is a multiple of the time unit. We achieve this property by rounding the travel time to the nearest larger multiple of the time unit in our implementation. Rounding up is a conservative method, however it always guarantees the solution of the time discretized model to be feasible in continuous time.

For the description purpose, we start with a single vehicle model.

9.2 The Single Vehicle Model

The model we propose for the IRP-CM is a vehicle flow formulation. We start by describing the network $G = (N, A)$, in which the vehicle flows.

N represents a set of nodes; a set of regular nodes $N_r = \{(i,t) : i \in \mathcal{P} \bigcup \mathcal{C}, t \in \{1, 2, \ldots, T\}\}$, where node (i,t) represents the possibility for the vehicle to visit site i at time t, and one dummy node $(0, T+1)$, which requires that the vehicle will end up somewhere after time T, i.e., $N = N_r \bigcup \{(0, T+1)\}$. Since the vehicle becomes available at location i^0 at time t^0, the vehicle starts its tour at node (i^0, t^0).

We denote by A the set of arcs. Arcs represent the vehicle's travel between two locations or waiting at a location. A travel arc $((i,t),(j,t+t_{ij}))$, where $(i,t) \in N_r$, $(j,t+t_{ij}) \in N_r$, and $i \neq j$, represents the vehicle traveling from location i to location j, starting at time t and ending at time $t + t_{ij}$. We use c_a to denote the cost of using arc $a \in A$. As defined before, the transportation cost associated with the travel between location i and j is c_{ij}. When $a = ((i,t),(j,t+t_{ij}))$, we set $c_a = c_{ij}$. A waiting arc $((i,t),(i,t+1))$, where $i \in \mathcal{P} \bigcup \mathcal{C}$ and $t \in \{1, 2, \ldots, T-1\}$, represents the vehicle waiting at location i from time t to time $t+1$ before it departs location i for another location. We set $c_a = 0$ for these arcs in the experimentation because the vehicle does not travel when using these arcs. However, we need to make the cost of using the waiting arc positive if there are penalties for the vehicle's waiting. The remaining arcs are $((i,T),(0,T+1))$ for all $i \in \mathcal{P} \bigcup \mathcal{C}$, and the cost of using this arc is 0. Note that, for each arc, the time component of its head node is strictly larger than that of its tail node. This implies G is directed acyclic.

The parameters $u_{(i,t)}$ and $p_{(j,t)}$ represent the usage of customer $i \in \mathcal{C}$ from time $t-1$ to time t and the production of plant $j \in \mathcal{P}$ from time $t-1$ to time t, respectively. Customer $i \in \mathcal{C}$ and plant $j \in \mathcal{P}$ have storage capacities C_i and C_j respectively.

The decision variable x_a associated with arc $a \in A$ takes the value 1 if the vehicle traverses a in the optimal solution and takes 0 otherwise. We denote by d_n a delivery or a pickup quantity at node $n \in N_r$. If n is a customer node, then d_n is a delivery quantity, and if n is a plant node, then d_n is a pickup quantity. Through these two sets of decision variables, we decide the vehicle's visit schedule (routes and time) and the delivery/pickup quantity for each visit. The following variables are implied quantities from these decisions.

We denote v^t to be the volume of product in the tank of the vehicle right after time t. We use v^0 to represent the initial volume of the product in its tank. For a node $n = (i,t) \in N_r$, we denote by I_n the inventory level of site i right after time t. We use $I_{(i,0)}$ to represent the initial inventory level of site i.

We can now formulate the time discretized model for IRP-CM. We start with the constraints first. The following constraints are commonly known as flow conservation constraints.

$$\sum_{\{a \in A:\ \text{head of } a \text{ is } n\}} x_a - \sum_{\{a \in A:\ \text{tail of } a \text{ is } n\}} x_a = 0,\ \forall n \in N_r \backslash (i^0, t^0),$$

$$\sum_{\{a \in A:\ \text{tail of } a \text{ is } (i^0,t^0)\}} x_a = 1,$$

$$\sum_{\{a \in A:\ \text{head of } a \text{ is } (0,T+1)\}} x_a = 1.$$

The first set of constraints imposes that if the vehicle arrives at one of the regular nodes then it has to leave that node. The second and the third constraints impose that the vehicle must start its tour at node (i^0, t^0) and end at the dummy node $(0, T+1)$.

A delivery or a pickup at node $n \in N_r$ can occur only when the vehicle visits node n. The following constraints impose this.

$$d_n \leq Q \sum_{\{a \in A:\ \text{tail of a is n}\}} x_a,\ \forall n \in N_r.$$

The following constraints represent the relation that the volume of the product in the tank of the vehicle at t is equal to the volume of the product in the tank at $t-1$ minus the volume of the delivery between $t-1$ and t plus the volume of the pickup between $t-1$ and t.

$$v^{t-1} - \sum_{\{n=(i,t)\ :\ n \in N_r, i \in \mathcal{C}\}} d_n + \sum_{\{n=(i,t)\ :\ n \in N_r, i \in \mathcal{P}\}} d_n = v^t,\ \forall t \in \{1, 2, ..., T\}.$$

At each node $n = (i,t) \in N_r$, we update the inventory level for site i.

$$I_{(i,t-1)} + d_n - u_{(i,t)} = I_{(i,t)},\ \forall i \in \mathcal{C}, \forall n = (i,t) \in N_r,$$
$$I_{(i,t-1)} - d_n + p_{(i,t)} = I_{(i,t)},\ \forall i \in \mathcal{P}, \forall n = (i,t) \in N_r,$$
$$I_{(i,t-1)} \geq u_{(i,t)},\ \forall i \in \mathcal{C}, \forall (i,t) \in N_r,$$
$$I_{(i,t-1)} \leq C_i - p_{(i,t)},\ \forall i \in \mathcal{P}, \forall (i,t) \in N_r.$$

The last two sets of constraints impose that none of the customers experience a stockout and none of the plants experience a venting respectively.

The following constraints are bound conditions.

$$x_a \in \{0,1\}\ \forall a \in A,$$
$$d_n \geq 0\ \forall n \in N_r,$$
$$0 \leq v^t \leq Q\ \forall t \in \{1, 2, ..., T\},$$
$$I_{(i,t)} \leq C_i,\ \forall i \in \mathcal{C}, \forall (i,t) \in N_r,$$
$$I_{(i,t)} \geq 0,\ \forall i \in \mathcal{P}, \forall (i,t) \in N_r,$$

The objective function of the model is

$$min \quad \sum_{a \in A} c_a x_a$$

This single vehicle model can be easily extended to the multiple vehicle model.

9.3 The Multiple Vehicle Model

Recall that $\mathcal{V}$ is a set of vehicles. We denote by $G^k(N, A^k)$ the network for vehicle $k \in \mathcal{V}$. Each vehicle k shares the node set N with other vehicles, but has its own arc set A^k. A^k is defined exactly the same as A in the single vehicle model. We define $A^{\mathcal{V}} = \bigcup_{k \in \mathcal{V}} A^k$. The number of inventory update constraints in the multiple vehicle model is the same as the number of those constraints in the single vehicle model because every vehicle uses the same N, which implies that the inventory update constraints link all vehicle activities. We use d_n^k and v_k^t to indicate vehicle k's pickup/delivery volume and the volume of the product in its tank respectively.

The formulation of the multiple vehicle model with the time discretization is

$$z = \quad min \sum_{k \in \mathcal{V}} \sum_{a \in A^k} c_a x_a \tag{18}$$

$s.t.$

$$\sum_{\{a \in A^k : \text{head of } a \text{ is } n\}} x_a - \sum_{\{a \in A^k : \text{tail of } a \text{ is } n\}} x_a = 0, \forall k \in \mathcal{V}, \forall n \in N_r \backslash (i_k^0, t_k^0), \tag{19}$$

$$\sum_{\{a \in A^k : \text{ tail of } a \text{ is } (i_k^0, t_k^0)\}} x_a = 1, \ \forall k \in \mathcal{V}, \tag{20}$$

$$\sum_{\{a \in A^k : \text{ head of } a \text{ is } (0, T+1)\}} x_a = 1, \ \forall k \in \mathcal{V}, \tag{21}$$

$$v_k^{t-1} - \sum_{\{n=(i,t) \ : \ i \in \mathcal{C}\}} d_n^k + \sum_{\{n=(i,t) \ : \ i \in \mathcal{P}\}} d_n^k = v_k^t, \ \forall t \in \{1, 2, ..., T\}, \forall k \in \mathcal{V}, \tag{22}$$

$$I_{(i,t-1)} + \sum_{k \in \mathcal{V}} d_n^k - u_{(i,t)} = I_{(i,t)}, \ \forall i \in \mathcal{C}, \forall n = (i,t) \in N_r, \tag{23}$$

$$I_{(i,t-1)} - \sum_{k \in \mathcal{V}} d_n^k + p_{(i,t)} = I_{(i,t)}, \ \forall i \in \mathcal{P}, \forall n = (i,t) \in N_r, \tag{24}$$

$$I_{(i,t-1)} \geq u_{(i,t)}, \ \forall i \in \mathcal{C}, \forall (i,t) \in N_r, \tag{25}$$

$$I_{(i,t-1)} \leq C_i - p_{(i,t)}, \ \forall i \in \mathcal{P}, \forall (i,t) \in N_r, \tag{26}$$

$$d_n^k \leq Q \sum_{\{a \in A^k:\ \text{tail of } a \text{ is } n\}} x_a,\ \forall k \in \mathcal{V},\ \forall n \in N_r, \tag{27}$$

$$x_a \in \{0,1\}\ \forall a \in A^k,\ \forall k \in \mathcal{V}, \tag{28}$$

$$d_n^k \geq 0\ \forall k \in \mathcal{V},\ \forall n \in N_r, \tag{29}$$

$$0 \leq v_k^t \leq Q\ \forall t \in \{1,2,...,T\},\ \forall k \in \mathcal{V}, \tag{30}$$

$$I_{(i,t)} \leq C_i,\ \forall i \in \mathcal{C},\ \forall (i,t) \in N_r, \tag{31}$$

$$I_{(i,t)} \geq 0,\ \forall i \in \mathcal{P},\ \forall (i,t) \in N_r. \tag{32}$$

$$\tag{33}$$

The constraints (19) - (21), or the flow conservation constraints, ensure that vehicle k tours from node (i_k^0, t_k^0) to node $(0, T+1)$. Constraints (22) update the volume of the product in each vehicle's tank at every t. Constraints (23) and (24) update inventory levels for customers and plants respectively. Constraints (25) ensure that customers do not run out of the product. Constraints (26) ensure that none of the plants experience a venting. Constraints (27) impose that we cannot make a delivery/pickup without vehicles. Constraints (28) - (32) are ordinary bound conditions. The objective (18) is to minimize total transportation costs over the planning horizon T.

Before the discussion of the solution techniques, we consider some variations of the model.

9.4 Variations

In this section, we discuss some problem characteristics that can be modeled with the time discretized model.

9.4.1 Stockout and venting

The objective defined in the problem description of IRP-CM is to minimize transportation costs over the planning horizon T, while trying to ensure that none of the customers experience a stockout. If stockouts are inevitable, then the time discretized model will

have no feasible solutions. Since we introduced limited storage capacities for plants in the time-discretized model, venting is conceivable. Especially for Argon distribution, venting is indeed a serious issue because the production of argon depends on the production of Oxygen. Incorporating stockouts and ventings into the time-discretized model is not a difficult task.

We denote by $S_{(i,t)}$ stockout amount for customer i from time $t-1$ to time t, and by $V_{(j,t)}$ the amount of venting at plant j from time $t-1$ to time t. Constraints (23), (24), (25), and (26) have to be modified to the following set of constraints.

$$
\begin{aligned}
& I_{(i,t-1)} + \sum_{k \in \mathcal{V}} d_n^k - u_{(i,t)} + S_{(i,t)} = I_{(i,t)},\ \forall i \in \mathcal{C},\ \forall n = (i,t) \in N_r, \\
& I_{(i,t-1)} - \sum_{k \in \mathcal{V}} d_n^k + p_{(i,t)} - V_{(i,t)} = I_{(i,t)},\ \forall i \in \mathcal{P},\ \forall n = (i,t) \in N_r, \\
& I_{(i,t-1)} + S_{(i,t)} \geq u_{(i,t)},\ \forall i \in \mathcal{C},\ \forall (i,t) \in N_r, \qquad (34) \\
& C_i - I_{(i,t-1)} + V_{(i,t)} \geq p_{(i,t)},\ \forall i \in \mathcal{P},\ \forall (i,t) \in N_r, \qquad (35) \\
& S_{(i,t)} \geq 0, \\
& V_{(i,t)} \geq 0.
\end{aligned}
$$

Constraints (34) ensure that the product delivered at time t cannot be used to reduce the amount of stockout occurring from $t-1$ to t. Similarly, constraints (35) ensure that we cannot reduce the amount of venting occurring from $t-1$ to t by picking up the product at t.

We have two objectives in this case. The first objective is minimizing total volume of stockout and venting; the second objective is minimizing total transportation costs. In order to handle both objectives appropriately, we introduce a two phase approach. In Phase 1, the objective function is

$$
z = \quad min \sum_{\{(i,t) \in N_r : i \in \mathcal{C}\}} S_{(i,t)} + \sum_{\{(i,t) \in N_r : i \in \mathcal{P}\}} V_{(i,t)}.
$$

If the optimal objective of Phase 1 is positive, then it indicates that a stockout, or venting, or both, are inevitable. If this is the case, it indicates that a vendor has to arrange

emergency schedules to guarantee no stockouts, or to reduce the amount of venting. After appropriate arrangements, we proceed to the second phase. The objective in Phase 2 is to minimize total transportation costs, while forcing all $S_{(i,t)}$ and $V_{(i,t)}$ to be zero.

9.4.2 Operating Mode/Delivery Mode

A customer may have the start and end time of its product usage on each day. A customer may also have time restrictions on deliveries. We denote the former by operating mode and the latter by delivery mode. Operating modes are not necessarily the same as delivery modes.

Operating modes are handled by appropriately setting $u_{(i,t)}$ in the time discretized model because the time discretized model does not rely on the assumption that customers have constant usage rates.

Delivery modes can be handled by forcing some variables to be zero. For each customer i, if customer i does not allow the product to be delivered at time t we can set $d_{(i,t)} = 0$.

9.4.3 Plant Breakdown

For the planning purpose, a breakdown of a plant has an impact only on the volume of production in that plant. We can handle plant breakdowns in the time-discretized model by appropriately setting $p_{(j,t)} = 0$, the production of plant $j \in \mathcal{P}$ from time $t-1$ to time t. This indicates again that the time discretized model does not rely on the assumption that plants have constant production rates.

9.4.4 Vehicle Maintenance Schedule

Suppose we have a maintenance schedule for a vehicle ζ. We assume that, according to the maintenance schedule, the vehicle ζ has to visit plant σ at time τ and it takes ω time units for its maintenance, based on the discretized time unit. Let $A_1^\zeta = \{((\sigma,t),(\sigma,t+1)) \in A^\zeta : \tau \leq t < \tau+\omega\}$. By forcing $x_a = 1$ for all $a \in A_1^\zeta$, we can achieve our goal; the vehicle ζ has to stay at facility σ for its maintenance. Let $A_2^\zeta = \{((i_1,t_1),(i_2,t_2)) \in A^\zeta : \tau \leq t_1 < \tau+\omega$ or $\tau < t_2 \leq \tau+\omega$ or $t_1 < \tau, t_2 > \tau+\omega\}$ and $A_3^\zeta = A_2^\zeta \backslash A_1^\zeta$. Every arc in A_2^ζ shares at least a moment in time interval $(\tau, \tau+\omega)$. The vehicle must use all arcs in A_1^ζ for its

maintenance. Thus it is easy to see that A_3^ζ can be removed from A^ζ because none of arcs in A_3^ζ can be used in a feasible solution.

CHAPTER X

SOLUTION TECHNOLOGY FOR THE TIME-DISCRETIZED MODEL

We devote this chapter to solution methodologies for the time-discretized model of the IRP-CM.

We start with several methods developed to reduce the problem size. These are important steps to solve the time-discretized model of IRP-CM because time discretization tends to make the size of a problem large. Then we discuss the specialized IP techniques devised to solve the reduced problem. These specialized techniques include strengthening the LP relaxation by a set of valid inequalities and specialized branching schemes.

Before discussing solution technologies in detail, we need to mention the minimum delivery quantities again. As we have adopted the use of minimum delivery quantities as part of our greedy heuristic, we also force the minimum delivery quantity requirement in our solution technology for the time-discretized model. We gave several reasons why we use minimum delivery quantities in the chapter on heuristics. In addition, since it is computationally prohibitive to solve a long-term IRP-CM with the time-discretized model, we approach it with a relatively short-term planning horizon. Decisions with a short-term planning horizon may lead to an undesirable situation in the next planning period. However, we can avoid this problem by using a minimum delivery quantity requirement.

10.1 Problem Size Reduction

Reducing the problem size is a crucial step in solving the time discretized model. In this section, we discuss how to decrease the problem size, while trying to maintain solution's quality.

10.1.1 Time Structure

A rolling horizon approach is often used to solve a long-term planning problem. The inventory routing problem is an example of such problem. In rolling horizon framework, we construct a distribution plan for the current planning period, but we implement only the decisions for the first few days. This implies that we need to decide the distribution plan precisely for the first few days. However, for the remaining days, outlines of the schedule may be sufficient because the role of decisions for these days is to capture the effects of decisions for the portion of the schedule that will be implemented, but decisions for these later days will be revisited when we move to the next planning period. That is why we can adopt different time units for the first few days and the next remaining days (a fine time unit for the near future, and a coarse time unit for the far future). The period, which has a coarse time unit in the current planning period, will have a fine time unit in the next planning period under the rolling horizon framework.

The problem size can be reduced dramatically by using multiple time units because the number of discretized time slots is directly related to the problem size.

Figure 18 shows a simple example with a 5 day planning horizon. A 1 hour time unit is used for the first 2 days, and a 2 hour time unit is used for the next 3 days. This reduces the number of time slots to $84(= 2 * 24 + 3 * 12)$, from 120 with a 1 hour time unit for the entire 5 days.

10.1.2 Network Structure

In the current network model, there might be some redundant nodes, arcs, or coverage of vehicles. In the following subsections, we discuss them.

10.1.2.1 Nodes and Delivery Periods

The economic fill level for a customer is defined by its storage capacity minus its minimum delivery quantity. Only when inventory drops below the economic fill level can we make a delivery. The period in which we can make deliveries is called the economic window. If the inventory level is not in the economic window, the minimum delivery quantity requirement

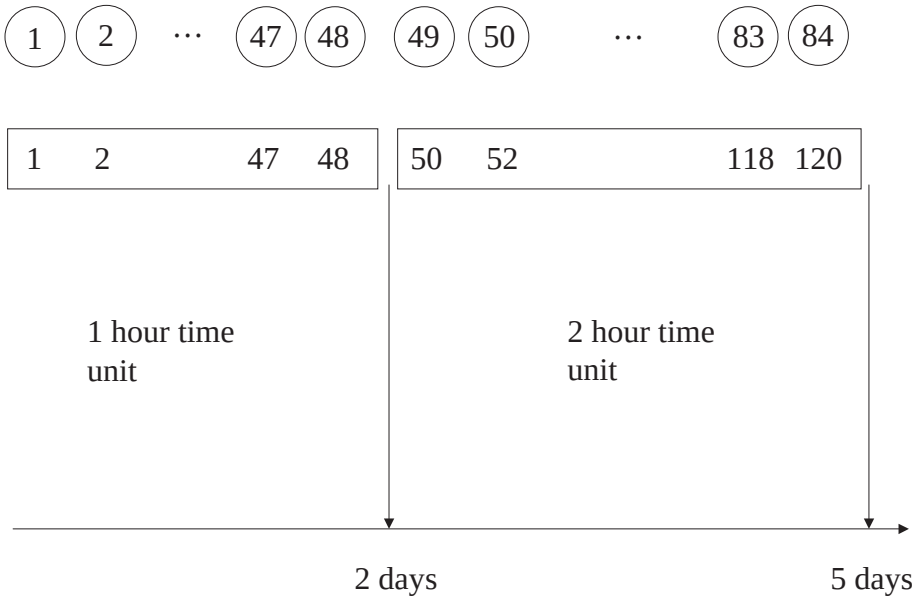
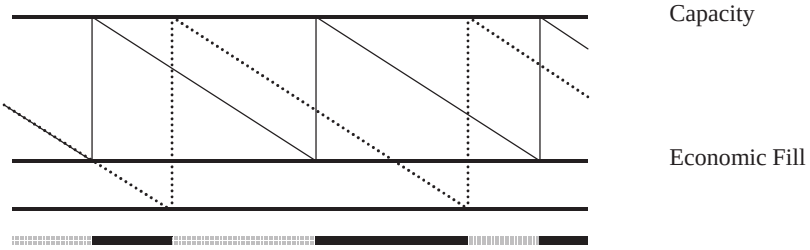


Figure 18: Example of Time structure

cannot be satisfied. It implies that a customer node for which its inventory level cannot be in its economic window can be removed from the network, while maintaining the solution quality. A non-delivery period is a time period in which the inventory level is surely above the economic fill level. In Figures 19 and 20, these two typical cases show how to figure out non-delivery periods and complementarily delivery periods.

Figure 19 illustrates a situation in which a customer cannot take a full truck load and Figure 20 illustrates a situation in which a customer's storage capacity is larger than the vehicle's tank capacity. The solid line represents the inventory level when the delivery is always made at the earliest possible time. This implies that it always takes the minimum delivery quantity. The dotted line represents the inventory level when the delivery time is the latest time without causing stockouts, and the delivery size is always the maximum delivery quantity it can take. It is not difficult to see that the inventory level of this customer is always above the economic fill level during the non-delivery periods. A node for which time is in the non-delivery periods can be removed from the model.

When customer's tank capacity < vehicle's tank capacity

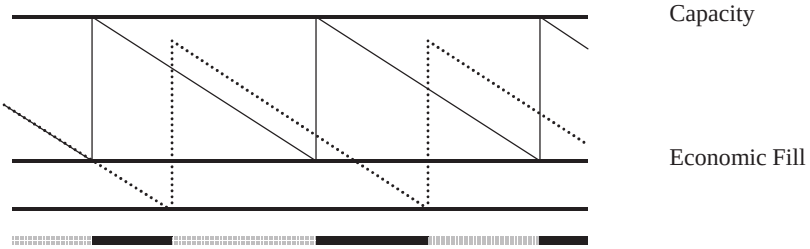


Non-delivery period (delete nodes in this period)

Delivery period (don't delete nodes in this period)

Figure 19: Storage Capacity with Less Than Truck Load

When customer's tank capacity > vehicle's tank capacity



Non-delivery period (delete nodes in this period)

Delivery period (don't delete nodes in this period)

Figure 20: Storage Capacity with More Than Truck Load

Suppose that customer i has an initial inventory level I_i^0, which is higher than its economic fill level $e_i = C_i - \underline{d_i}$, where C_i is a storage capacity and $\underline{d_i}$ is the minimum delivery quantity for customer i. Let $\overline{d_i} = min\{C_i, Q\}$. Then $\overline{d_i}$ is the maximum delivery quantity that customer i can take. The first delivery period for customer i starts at time $t^s_{D_i^1} = \frac{I_i^0 - e_i}{u_i}$ and ends at time $t^e_{D_i^1} = \frac{I_i^0}{u_i}$. The second delivery period starts at time $t^s_{D_i^2} = t^s_{D_i^1} + \frac{d_i}{u_i}$ and ends at time $t^e_{D_i^2} = t^e_{D_i^1} + \frac{\overline{d_i}}{u_i}$. The last delivery period satisfies either $t^s_{D_i^l} < T, t^e_{D_i^l} \geq T$ or $t^e_{D_i^l} < T, t^s_{D_i^{l+1}} \geq T$. For the time being, we assume that each delivery period is not being overlapped with its next delivery period, i.e., $t^e_{D_i^k} < t^s_{D_i^{k+1}}, \forall k = 1, 2, ..., l-1$. In case of $I_i^0 \leq e_i$ and $I_i^0 + \underline{d_i} > e_i$, $t^s_{D_i^1} = 0$ and $t^s_{D_i^2} = \frac{I_i^0 + \underline{d_i} - e_i}{u_i}$. The others remain the same as before. A delivery period is called closed if its end time is less than the planning horizon T. Otherwise, it is called an open delivery period. It is easy to see that only one delivery is possible in a delivery period because of the minimum delivery quantity. A delivery with a larger size than its minimum delivery quantity keeps the inventory level above the economic fill level, at least until the start time of next delivery period. In order to guarantee no stockouts, we need to schedule a customer visit during a closed delivery period. However, the necessity of a customer visit during an open delivery period depends on the sizes of the former deliveries. So we may, or may not, need to schedule a customer visit during an open delivery period. We use these observations in the development of a branch-and-cut algorithm.

It is not difficult to see that we can make a delivery at most once in a non-overlapping delivery period. This observation plays an important role when we develop specialized IP techniques for the IRP-CM later in this chapter. In the data set provided by Praxair, there are no customers who have an overlapping delivery period when $T = 5$ *days*. However, delivery periods can be overlapped if we adopt a longer planning horizon T, because the increase in the end time of two consecutive delivery periods for a customer can be larger than an increase in the start time as time increases. Moreover, it is possible that $t^s_{D_i^1}$, the start time of the first delivery period for customer i, can be the same as $t^s_{D_i^2}$, the start time of the second delivery period for customer i, if $I_i^0 + \underline{d_i} \leq e_i$ (e.g., $I_i^0 + 2\underline{d_i} \leq C_i$). To avoid the overlapping cases, we force $t^s_{D_i^{k+1}} = t^e_{D_i^k} + \epsilon$ if $t^s_{D_i^{k+1}} \leq t^e_{D_i^k}$, where ϵ is a small positive

value. By doing this, we may lose some solution qualities, but we believe that the impact of this is negligible in practice because inventory routing problems are usually solved in a rolling horizon framework. Even if there exists an overlapping delivery period, it only tends to occur at a later part of the current planning period, for which the schedule will not be implemented in the rolling horizon framework.

10.1.2.2 Arcs

Long arcs are costly and less likely to occur in an optimal solution. We use the following arc rules to remove some of these long arcs.

An arc rule determines the accessibility between two sites in the network. In our model, we use the (α, β_c)-*rule* to determine the accessibility between customers, and the β_p-*rule* between a customer and a plant.

Under the (α, β_c)-*rule*, if the travel distance from customer j to customer i is less than or equal to α, then customer i is accessible from customer j. If the number of accessible customers to customer i is less than β_c, then we add the other closest customers until it is equal to β_c. Finally, if customer j is accessible from customer i, then customer i is also accessible from customer j.

Let $\underline{d_i}$ and $\underline{d_j}$ be the minimum delivery quantities for customer i and j respectively. Even though customer i and j are accessible from each other, if $\underline{d_i} + \underline{d_j} > Q$, then we do not add arcs between customer i and j. To satisfy the minimum delivery quantity requirement, a vehicle cannot visit these two customers without an additional pickup.

Under the β_p-*rule*, for each customer i, we find β_p closest plants. Customer i is accessible from these plants and vice versa.

We do not allow any arcs between plants, i.e., there is no accessibility between plants. Sometimes this does happen in practice. A vehicle visits a supply facility to deliver a certain amount of product. Even though we do not consider this in order to maintain the simplicity of the model, it is not difficult to allow deliveries to plants in the model. By allowing d_n to take negative values where n is a plant node, deliveries to plants will be possible. If no vehicle visits node n, then d_n has to remain 0.

The choice of parameters impacts both the solution quality and time. Reasonable values for α, β_c, and β_p depend on instances.

Once the accessibility lists for customers and plants are ready, we define an arc set A based on the lists and nodes in the delivery periods. The arc set A consists of the following elements. A waiting arc $((j,t),(j,t+1))$ indicates waiting at plant j, where $j \in \mathcal{P}$ and $t \in \{1,2,\ldots,T-1\}$. $((j,T),(0,T+1))$ for all $j \in \mathcal{P}$ are also in A as before. For each delivery period of customer i, $((i,t),(i,t+1))$ indicates an arc waiting at customer i, where $t \in \{\lceil t^s_{D_i} \rceil, ..., \lfloor t^e_{D_i} \rfloor - 1\}$. The last node in a delivery period has to be accessible to the dummy node. So $((i, \lfloor t^e_{D_i} \rfloor),(0,T+1))$ is also in A for each delivery period of customer i. Note that customer node $(i,t) \in N$ indicates that t is in one of delivery periods of customer i. Suppose that i and j are accessible. $((i,t),(j,t+t_{ij}))$ is an arc between two sites, where $(i,t) \in N$, $(j,t+t_{ij}) \in N$, and $i \neq j$. Suppose that customer i and customer j are accessible and $\lfloor t^e_{D^q_i} \rfloor + t_{ij} < \lceil t^s_{D^r_j} \rceil$. Then $((i, \lfloor t^e_{D^q_i} \rfloor),(j, \lceil t^s_{D^r_j} \rceil))$ is an arc from the last node in the q^{th} delivery period of customer i to the first node in the r^{th} delivery period of customer j.

After generating A, we denote A^k to be a set of arcs for vehicle k.

10.1.2.3 Plant-Vehicle Alignments

A vehicle is allowed to visit any plant and any customer in the IRP-CM. By limiting this, the problem size can be reduced further. We call this method plant-vehicle alignments.

We define $\mathcal{P}^k \subseteq \mathcal{P}$ to be a set of plants that vehicle k is allowed to visit. Then we define $\mathcal{C}^k$ to be the set of customers accessible from at least one element of $\mathcal{P}^k$. This implies that the network for vehicle k is now $G^k(N^k, A^k)$ where $N^k = \{(i,t) \in N : i \in \mathcal{P}^k \bigcup \mathcal{C}^k \bigcup \{0\}\}$ and $A^k = \{a \in A^k : \text{both ends of } a \text{ are in } N^k\}$. It is important to define $\mathcal{P}^k$ appropriately so that each plant is aligned with enough vehicles. This should be determined based on the plant's production rate, distances from other plants, and vicinity of customers.

10.2 Specialized IP Techniques

10.2.1 Delivery Cover Inequalities

Due to the minimum delivery quantities, customers with sum of their minimum quantities exceeding Q cannot be combined in a tour. Let F be a subset of $\mathcal{C}$. Then $\lceil \sum_{i \in F} \underline{d_i}/Q \rceil$

is an obvious lower bound for the number of vehicles needed to make a delivery to the customers in F at least once. This simple observation gives us the valid inequalities named delivery cover inequalities. Before detailed explanations for delivery cover inequalities, we need additional concepts and constraints to incorporate minimum delivery quantities into the time discretized model. Through these concepts, we can concentrate our efforts on making high level decisions first, such as routes, visits, and deliveries, without considering specific timing.

A *supernode* $\mathbf{s}_u^k$ represents the set of nodes in the k^{th} delivery period for customer u. If the k^{th} delivery period for customer u is an open delivery period, $\mathbf{s}_u^k$ is called an open supernode. Otherwise, it is a closed supernode. $\mathcal{S}^o$ and $\mathcal{S}^c$ represent the set of all open supernodes and the set of all closed supernodes respectively (and we define $\mathcal{S} = \mathcal{S}^o \bigcup \mathcal{S}^c$). Four types of *superarcs* exist. A superarc $(\mathbf{s}_u^k, \mathbf{s}_v^{k'})$ represents a set of arcs going from a node in supernode $\mathbf{s}_u^k$ to a node in supernode $\mathbf{s}_v^{k'}$. A superarc $(\mathbf{s}_u^k, j)$ represents a set of arcs going from a node in supernode $\mathbf{s}_u^k$ to a node of plant j. A superarc $(j, \mathbf{s}_u^k)$ represents a set of arcs going from a node of plant j to a node in supernode $\mathbf{s}_u^k$. A superarc $(\mathbf{s}_u^k, (0, T+1))$ represents a set of arcs going from a node in supernode $\mathbf{s}_u^k$ to the dummy node $(0, T+1)$. Figure 21 shows these four types of superarcs.

We denote by $\delta^+(\mathbf{s}_i^k)$ the set of arcs which are elements of outgoing superarcs from supernode $\mathbf{s}_i^k$. Then the additional constraints are

$$\sum_{n \in \mathbf{s}_i^k} d_n \geq \underline{d_i}, \ \forall \mathbf{s}_i^k \in \mathcal{S}^c, \tag{36}$$

$$\sum_{a \in \delta^+(\mathbf{s}_i^k)} x_a = 1, \ \forall \mathbf{s}_i^k \in \mathcal{S}^c, \tag{37}$$

$$\sum_{n \in \mathbf{s}_i^k} d_n \geq \underline{d_i} \sum_{a \in \delta^+(\mathbf{s}_i^k)} x_a, \ \forall \mathbf{s}_i^k \in \mathcal{S}^o, \tag{38}$$

$$\sum_{a \in \delta^+(\mathbf{s}_i^k)} x_a \leq 1, \ \forall \mathbf{s}_i^k \in \mathcal{S}^o. \tag{39}$$

Constraints (36) and (37) ensure that a closed supernode is visited exactly once and the delivery size is greater than or equal to the minimum delivery quantity associated with the supernode. Constraints (38) and (39) ensure that a vehicle can visit an open supernode once

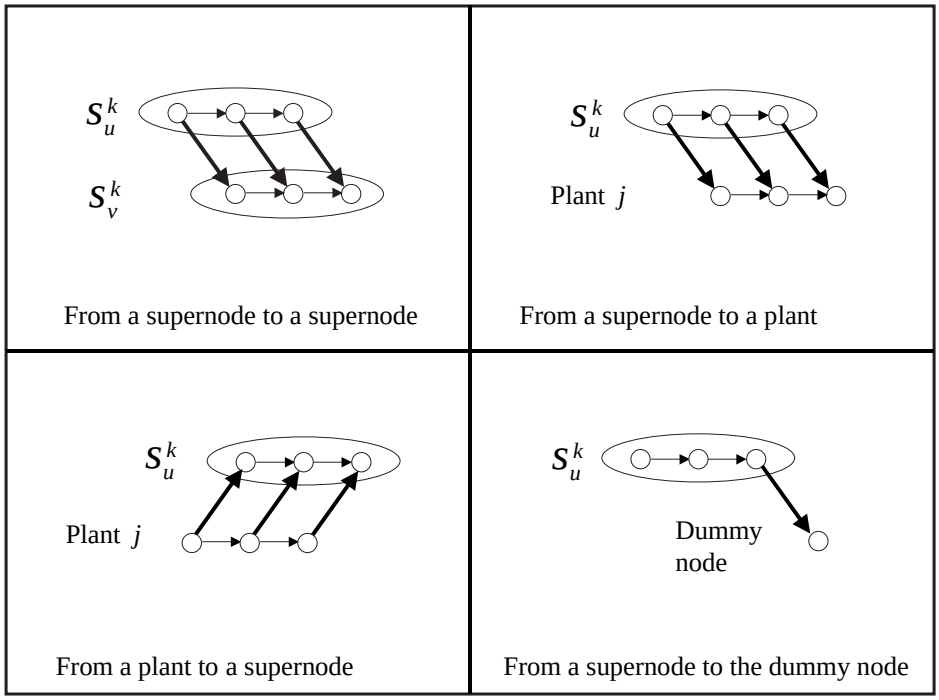


Figure 21: Four types of superarcs

and, if it does, it delivers more than or equal to the minimum delivery quantity associated with the supernode. The time discretized model with these constraints allows us to use delivery cover inequalities.

For the description of delivery cover inequalities, we will also use the following notations. Suppose $H \subseteq \mathcal{S}$. We define $\underline{d}(H) = \sum_{\mathbf{s}_i^k \in H} \underline{d_i}$. We denote by $\gamma(H)$ the set of arcs that are elements of superarcs with both the tail and the head supernodes in H. For a subset A of the arc set $A^{\mathcal{V}}$, we define $x(A) = \sum_{a \in A} x_a$. If an optimal solution for the LP relaxation is $\hat{x}$, then $\hat{x}(A) = \sum_{a \in A} \hat{x}_a$. The definition for delivery cover inequalities is as follows.

For any subset $H \neq \emptyset$ of the supernode set $\mathcal{S}$, $x(\gamma(H)) \leq |H| - \left\lceil \frac{\underline{d}(H)}{Q} \right\rceil$ is a valid inequality because $\left\lceil \frac{\underline{d}(H)}{Q} \right\rceil$ is a lower bound on the number of vehicle tours that visit all supernodes in H. This is what we call a delivery cover inequality.

This inequality may be strengthened by replacing the right hand side with the optimal value of the *Bin Packing Problem* (BPP), with bin size Q and item size $\underline{d_i}$ for each supernode in H. However, BPP is a NP-hard. In the separation heuristics below, we use $\left\lceil \frac{\underline{d}(H)}{Q} \right\rceil$ for

the sake of computational tractability.

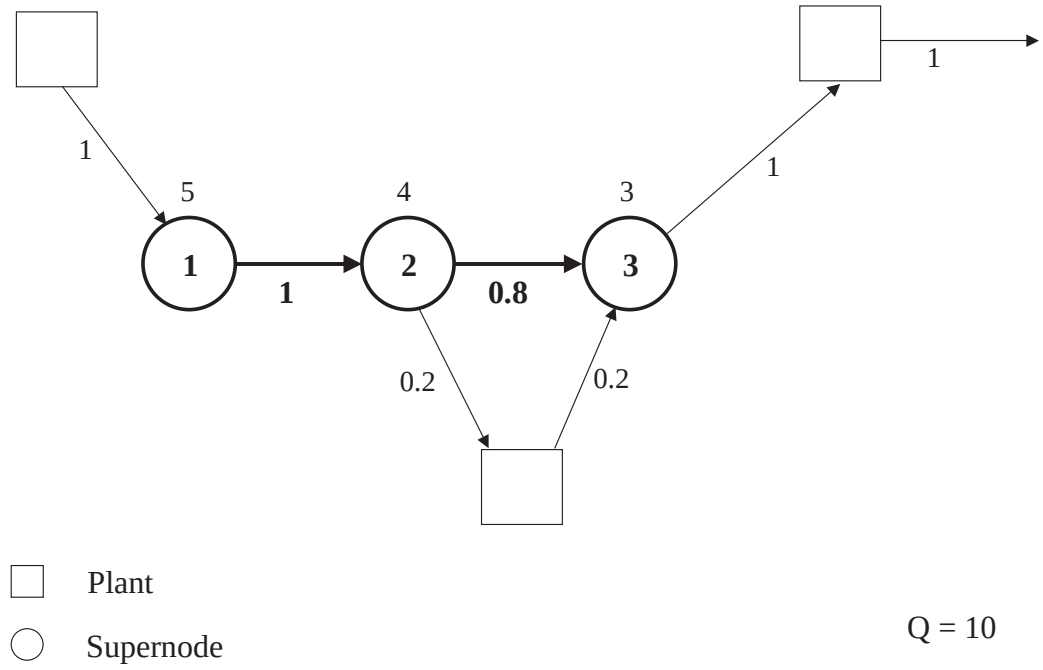


Figure 22: Simple Example of Delivery Cover Inequality Violated by $\hat{x}$

Figure 22 shows a simple example of a delivery cover inequality violated by $\hat{x}$. If we define a set of supernodes $H = \{1, 2, 3\}$, then $\underline{d}(H) = 5 + 4 + 3 = 12$, $\hat{x}(\gamma(H)) = 1 + 0.8 = 1.8$. Since the tank capacity of the vehicle is 10, which is less than $\underline{d}(H)$, the vehicle cannot visit these three supernodes consecutively in a tour without an additional pickup. That is, at least two vehicle tours are necessary to service these three supernodes. This induces $x(\gamma(H)) \leq |H| - \left\lceil \frac{\underline{d}(H)}{Q} \right\rceil$, a valid inequality which is violated by $\hat{x}$ because $\hat{x}(\gamma(H)) = 1.8 > 3 - 2 = 1$.

Delivery cover inequalities are very similar to the well-known capacity constraints of the *Capacitated Vehicle Routing Problem* (CVRP). A notable difference between the capacity constraints and the delivery cover inequalities is the existence of open supernodes. In the CVRP, every node is visited exactly once. On the other hand, an open supernode may not

be visited in the optimal solution. It is not difficult to see that the capacity constraints can be viewed as a special case of the delivery cover inequalities. A large body of literature on the branch-and-cut approaches for the CVRP exists; Naddef and Rinaldi [29], Ralphs et al. [31] and Lysgaard et al. [27] contain overviews of the recent major research activities in this area. The separation of the capacity constraints is known to be NP-complete (see [7]). Thus we use heuristics for the separation of delivery cover inequalities.

10.2.2 Separation Heuristics

Two heuristics are used to determine delivery cover inequalities violated by $\hat{x}$. The first heuristic is the *integer connected components heuristic*, and the second is called the *connected components heuristic*. The connected components heuristic is used only when the integer connected components heuristic fails to return a violated delivery cover inequality.

Let $G_{\hat{x}}$ and $G_{\hat{x}=1}$ be the networks supported by the arcs, whose associated $\hat{x}$ values are strictly positive and 1 respectively. $G_{\hat{x}=1}$ is a subnetwork of $G_{\hat{x}}$. If a node in supernode $\mathbf{s}$ is connected to a node in supernode $\mathbf{s}'$, then we say that supernode $\mathbf{s}$ and $\mathbf{s}'$ are connected.

The integer connected components heuristic is inspired by the simple example of Figure 22. The integer connected components heuristic begins with constructing the maximal connected supernode components in $G_{\hat{x}=1}$. For each supernode component H, find a supernode $\mathbf{s} \notin H$ which is connected to H in $G_{\hat{x}}$. If $\hat{x}(\gamma(H \bigcup \{\mathbf{s}\})) > |H| + 1 - \left\lceil \frac{d(H \bigcup \{\mathbf{s}\})}{Q} \right\rceil$, then a violated delivery cover inequality is found.

It is easy to see that the integer connected components heuristic detects the delivery cover inequality that is violated by $\hat{x}$ in Figure 22. In Figure 22, $H = \{1, 2\}$ is a maximal connected supernode component in $G_{\hat{x}=1}$. Since supernode 3 is connected to H of $G_{\hat{x}}$, the integer connected components heuristic checks the supernode set $H \bigcup \{3\} = \{1, 2, 3\}$ to see whether the delivery cover inequality associated with this supernode set is violated by $\hat{x}$. That is, $\hat{x}(\gamma(\{1, 2, 3\})) = 1.8 > 2 + 1 - \left\lceil \frac{d(\{1,2,3\})}{10} \right\rceil = 3 - \lceil \frac{5+4+3}{10} \rceil = 1$. This indicates the delivery cover inequality associated with the supernede set {1,2,3}, $\hat{x}(\gamma(\{1, 2, 3\})) \leq 1$ is violated by $\hat{x}$.

If the integer connected components heuristic fails to determine a violated inequality, we

apply the connected components heuristic. We modify the connected components heuristic for the CVRP appearing in [31] appropriately, so that it can be used to produce violated delivery cover inequalities. The connected components heuristic begins with constructing the maximal connected supernode components in $G_{\hat{x}}$. For each supernode component H, the heuristic checks if $\hat{x}$ violates the delivery cover inequality associated with H. If the violation is detected, we add H to $\mathcal{H}$. Otherwise, we find a supernode $\mathbf{s} \in H$ in a specific way, so that the chance of violation of $H \backslash \{\mathbf{s}\}$ gets bigger than that of H. If such a node exists, we set $H \leftarrow H \backslash \{\mathbf{s}\}$ and check the violation again. Unless the delivery cover inequality violated by $\hat{x}$ is found, we continue this process until no such supernode exists. The connected components heuristic for the delivery cover inequalities is presented in Algorithm 4. The input for the connected components heuristic is $G_{\hat{x}}$ and its output is a set of delivery cover inequalities violated by $\hat{x}$. The output delivery cover inequalities are determined by $\mathcal{H}$.

Algorithm 4 Connected Components Heuristic

Find the supernode sets $H_1, \ldots, H_r$ of the maximal connected supernode components of $G_{\hat{x}}$
for $i = 1, 2, \ldots, r$ **do**
  if $\hat{x}(\gamma(H_i)) > |H_i| - \left\lceil \frac{\underline{d}(H_i)}{Q} \right\rceil$ **then**
    Add H_i to $\mathcal{H}$.
  else
    while $\exists \mathbf{s} \in H_i$ such that $\left\lceil \frac{\underline{d}(H_i)}{Q} \right\rceil = \left\lceil \frac{\underline{d}(H_i \backslash \{\mathbf{s}\})}{Q} \right\rceil$ and $|H_i| - 1 - \left\lceil \frac{\underline{d}(H_i \backslash \{\mathbf{s}\})}{Q} \right\rceil - \hat{x}(\gamma(H_i \backslash \{\mathbf{s}\})) < |H_i| - \left\lceil \frac{\underline{d}(H_i)}{Q} \right\rceil - \hat{x}(\gamma(H_i))$ **do**
      if $\hat{x}(\gamma(H_i \backslash \{\mathbf{s}\})) > |H_i| - 1 - \left\lceil \frac{\underline{d}(H_i \backslash \{\mathbf{s}\})}{Q} \right\rceil$ **then**
        Add $H_i \backslash \{\mathbf{s}\}$ to $\mathcal{H}$
        Break while
      else
        $H_i \leftarrow H_i \backslash \{\mathbf{s}\}$
      end if
    end while
  end if
end for

We apply the connected components heuristic only when the integer connected components heuristic fails to determine a violated inequality. Even though we can apply

the connected component heuristic first, it is better to use the integer connected components heuristic first because it is faster. Figure 23 displays an example in which the integer connected components heuristic fails to find a delivery cover inequality violated by $\hat{x}$, but the connected components heuristic finds one. The integer connected components heuristic constructs maximal integer connected supernode components, $\{1,2\}$ and $\{3,4\}$. Since delivery cover inequalities determined by $\{1,2,3\}$ and $\{2,3,4\}$ are not violated by $\hat{x}$, the integer connected components heuristic fails to find a cut. The connected components heuristic constructs $H = \{1,2,3,4\}$ as the maximal connected supernode component. The delivery cover inequality determined by H itself is violated because $\hat{x}(\gamma(\{1,2,3,4\})) = 2.5 > 4 - \left\lceil \frac{d(\{1,2,3,4\})}{10} \right\rceil = 4 - \lceil \frac{5+3+2+5}{10} \rceil = 2$.

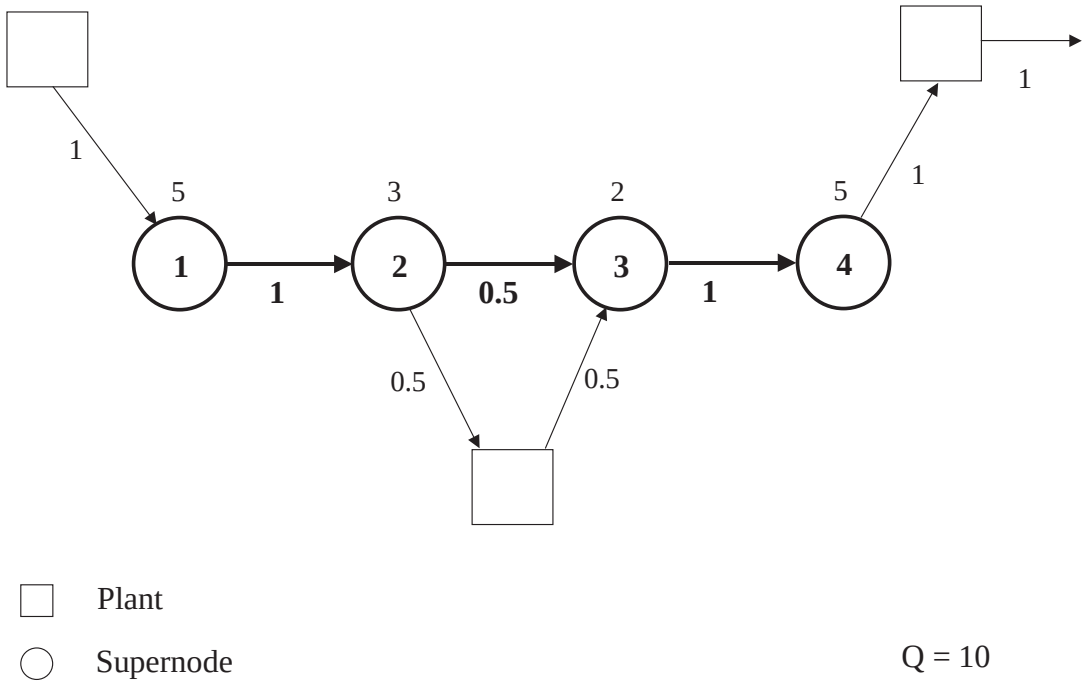


Figure 23: Simple Example for the Connected Component Heuristic

10.2.3 Branching Rules

Branching is also a critical component of our branch-and-cut algorithm. We have experimented with three branching rules.

The first rule is branching with MIP solver default settings. The second rule is branching on a fractional x_a variable with priority based on the time of its tail node. The earlier the time of the tail node it has, the higher priority it takes. The hope of this branching is that fixing an earlier part of the schedule makes the rest of the schedule easy to fall into place.

The last branching rule is branching on superarcs. $x_a = 1$ fixes many other x variables to zeros. On the other hand, $x_a = 0$ hardly fixes other variables to either of their upper and lower bounds. So branching on x_a tends to make the branching tree unbalanced. When we branch on superarcs, we find a superarc, let's say $(\mathbf{s}, \mathbf{s}')$, whose value $\hat{x}(\gamma\{\mathbf{s}, \mathbf{s}'\})$ is close to 0.5. Then we split the solutions into those in which $\sum_{a \in (\mathbf{s},\mathbf{s}')} x_a = 1$ and those in which $\sum_{a \in (\mathbf{s},\mathbf{s}')} x_a = 0$. The time-discretized model tries to decide routes, and the time and size of deliveries congruently. The super arc branching leads the MIP solver to consider routes for deliveries with higher priority, and our expectation is that branching on superarcs makes the branching tree more balanced.

10.2.3.1 Further Issue

Another possible branching rule, even though we have not experimented with it, is branching on $\gamma(H)$, where H is a subset of $\mathcal{S}$. Suppose $|H| - \left\lceil \frac{d(H)}{Q} \right\rceil - 1 < \hat{x}(\gamma(H)) < |H| - \left\lceil \frac{d(H_i)}{Q} \right\rceil$. Then we can split the solutions into those in which $x(\gamma(H)) = |H| - \left\lceil \frac{d(H_i)}{Q} \right\rceil$ and those in which $x(\gamma(H)) \leq |H| - \left\lceil \frac{d(H_i)}{Q} \right\rceil - 1$. To implement this branching, we need to consider how to determine a high quality H fast. The size of H, with respect to the impact on speed, may be another issue for the implementation.

CHAPTER XI

COMPUTATIONAL RESULTS

To analyze the branch-and-cut algorithm presented in the previous chapter, we introduce two more data sets (derived from the base instance used in Chapter 8). The first data set of 8 small instances ($s1$, $s2$, ..., $s8$) involves 2 plants, 51 customers (around these plants), and 2 vehicles. We randomly generated the initial inventory levels for the customers at the start of the planning period. This set of instances is used to analyze the performance of the various settings of the branch-and-cut algorithm. The second data set of 6 medium size instances ($m1$, $m2$, ..., $m6$) involves 3 plants, 3 vehicles, and about 100 customers. Again, we randomly generated the initial inventory levels for the customers at the start of the planning period. This set of instances is used to compare RGH with the time discretized MIP. Since RGH focuses on the model without finite storage capacities for plants, we assume that each plant has an infinite storage capacity in all instances. CPLEX 9.0 was used as a MIP solver. The experiments were run on a SUN 280R machine with 900MHz UltraSparc-III-Cu CPU.

The planning horizon used in the experiments is 5 days. We use different time units between the first 2 days and the remaining 3 days. A time unit (a, b) represents that the time unit for the first 2 days is a hours and the time unit for the next 3 days is b hours. We use the following base settings for the branch-and-cut algorithm, unless specifically stated otherwise. The time unit is $(2, 4)$. For the (α, β_c)-$rule$, we use $(300, 5)$. We do not utilize the β_p-$rule$ and plant-vehicle alignments for these experiments. That is, each customer can be serviced from all plants by all vehicles. Superarc branching is a part of the base settings. Delivery cover inequalities are the only cuts added to the branch-and-cut algorithm, which implies that CPLEX cuts are turned off in the base settings. Even though there are many ways to set the frequency of cut generations (e.g. using skip factor, using depth of the node in the tree, or using combinations of these), the separation heuristics are applied at the first 20 nodes and then every 20 nodes in the default setting. This default setting just represents

the frequency of cut generations positioned somewhere between root node only and every node.

In the following section, we provide some experimental results to compare these base settings with many other options.

11.1 Various Settings for the Branch-and-Cut Algorithm

In this section, we solve small instances to evaluate the effects of various settings and to determine appropriate settings for the branch-and-cut algorithm.

11.1.1 Time Discretization

We experimented with various time units to see how sensitive our instances are to the time discretization. The results are presented in Table 9. The last two columns show the percentage decrease in optimal objective values and the percentage increase in CPU time when switching from the $(2, 4)$ time unit to the $(1, 1)$ time unit. Out of 8 instances, 4 instances produced the same optimal objective values (s1, s2, s6, and s8). The largest percentage decrease is 3.99% (instance s7), and the average percentage decrease in the optimal objective value is 1.33%. On the other hand, the branch-and-cut algorithm with the $(1, 1)$ time unit took more than 13 times the CPU time then the branch-and-cut algorithm with the $(2, 4)$ time unit. For our instances, the time unit $(2, 4)$ works reasonably well.

As a rule of thumb, the time unit has to be small enough in order for the delivery period of a customer to contain at least several nodes in the network. The average length of the first delivery period for each customer in our instances is about one and half days.

11.1.2 Delivery Cover Inequalities

Table 10 demonstrates the effects of the deliver cover inequalities as cuts (DC cuts) in the branch-and-cut algorithm. The columns headed *Cplex cuts only* give the results with the Cplex default cuts, without delivery cover inequalities. The columns headed *DC cuts only* give the results with delivery cover inequalities, without Cplex default cuts. The columns headed *Cplex cuts and DC cuts* give the results with Cplex default cuts and delivery cover inequalities. The columns labeled *time (sec.)*, *B&C nodes*, *Cplex cuts*, and *DC cuts* give the

Table 9: Comparison of Various Time Units

	(2,4)		(2,2)		(1,2)		(1,1)		quality decrease from (2,4) to (1,1)	time increase from (2,4) to (1,1)
	obj^*	time	obj^*	time	obj^*	time	obj^*	time		
s1	1,474.1	7	1,474.1	36	1,474.1	41	1,474.1	393	0.00%	5717.3%
s2	1,540.9	249	1,540.9	730	1,540.9	1,975	1,540.9	3,537	0.00%	1322.0%
s3	1,730.4	74	1,730.4	234	1,716.9	679	1,716.9	1,149	0.78%	1456.7%
s4	1,148.9	259	1,148.9	296	1,105.4	1,315	1,105.4	2,803	3.78%	981.8%
s5	1,519.8	124	1,508.4	396	1,508.4	416	1,488.3	1,611	2.07%	1199.3%
s6	1,619.3	29	1,619.3	58	1,619.3	231	1,619.3	759	0.00%	2473.9%
s7	1,852.2	81	1,792.5	250	1,778.3	328	1,778.3	1,156	3.99%	1330.0%
s8	1,161.3	8	1,161.3	12	1,161.3	55	1,161.3	58	0.00%	659.5%
avg	1,505.8	104	1,497.0	252	1,488.1	630	1,485.6	1,433	1.33%	1280.8%

total CPU time, the number of $B\&C$ tree nodes, the number of Cplex cuts generated, and the number of delivery cover inequalities generated, respectively. The results in Table 10 clearly demonstrate the benefits of delivery cover inequalities. In terms of the average CPU time, the $B\&C$ with DC cuts only (= 104 seconds) performed slightly better than the $B\&C$ with Cplex cuts and DC cuts (= 124 seconds). We experienced that the difference becomes larger with a larger size of instances. The results with larger instances are presented in Table 11.

Table 10: The Effect of Delivery Cover Inequalities

Instance	Cplex cuts only			DC cuts only			Cplex cuts and DC cuts			
	time (sec.)	$B\&C$ nodes	Cplex cuts	time (sec.)	$B\&C$ nodes	DC cuts	time (sec.)	$B\&C$ nodes	Cplex cuts	DC cuts
$s1$	39	285	63	7	16	5	4	1	47	5
$s2$	1,161	3,945	230	249	340	25	231	185	235	25
$s3$	58	96	64	74	93	9	16	6	38	6
$s4$	38,848	88,697	374	259	261	23	458	516	239	20
$s5$	35,159	80,129	207	124	67	21	204	120	103	21
$s6$	63	245	165	29	47	5	28	25	46	7
$s7$	2,893	18,320	392	81	145	14	50	56	102	10
$s8$	75	318	94	8	2	4	3	1	13	3
average	9,787	24,004	199	104	121	13	124	114	103	12

In 5 out of 6 instances, the $B\&C$ with DC cuts found optimal solutions faster than the $B\&C$ with Cplex cuts and DC cuts. In terms of the average CPU time, the $B\&C$ with

DC cuts only (= 175,039 seconds) outperformed the $B\&C$ with Cplex cuts and DC cuts (= 324,486 seconds).

Table 11: The Effect of Delivery Cover Inequalities with larger instances

Instance	DC cuts only			Cplex cuts and DC cuts			
	time (sec.)	$B\&C$ nodes	DC cuts	time (sec.)	$B\&C$ nodes	Cplex cuts	DC cuts
$m1$	183,543	12,794	243	255,672	11,127	1,321	220
$m2$	47,068	4,892	119	72,688	4,035	1,007	134
$m3$	124,727	11,449	180	109,854	8,330	776	126
$m4$	143,278	20,472	204	310,797	29,010	949	280
$m5$	523,286	35,364	202	1,118,620	64,806	1,591	211
$m6$	28,330	5,380	114	79,286	10,138	954	137
average	175,039	15,059	177	324,486	21,241	1,100	185

We implemented two separation heuristics to find the DC cuts. In our default setting for $B\&C$, the connected components heuristic is used only when the integer connected components heuristic fails to return a delivery cover inequality. Table 12 shows the value of using two separation heuristics together. The columns headed Heuristic 1 gives the results when only the integer connected components heuristic is used as a separation heuristic. The columns headed Heuristic 2 gives the results when only the connected components heuristic is used as a separation heuristic. The columns headed Heuristic 3 gives the results with our default setting. In terms of average CPU time, $B\&C$ with two separation heuristics together, solved instance problems to optimality more quickly than the others. The numbers of DC cuts with different approaches are not significantly different, and the average $B\&C$ tree size with two heuristics together (121) is about half of the others (245 and 233). This shows an advantage of using two heuristics together.

Table 13 illustrates how often the DC cut generation routine needs to be called in the $B\&C$. The columns headed *Root node* give the results when the DC cuts are generated only at root node. The columns headed *Every 20* give the results when the DC cuts are generated first 20 nodes and then every 20 nodes. The columns headed *All nodes* give the results when the DC cuts are generated at every node. As in Table 13, the DC cuts need to be generated at child nodes of the $B\&C$ tree. In terms of the average CPU time, the $B\&C$ with DC cuts at all nodes (=140 seconds) takes longer than the $B\&C$ with DC cuts

Table 12: Comparison of Separation Heuristics

Instance	Heuristic 1			Heuristic 2			Heuristic 3			
	time (sec.)	MIP nodes	DC cuts	time (sec.)	MIP nodes	DC cuts	time (sec.)	MIP nodes	DC cuts	
									from H1	from H2
$s1$	7	16	5	10	15	13	7	16	5	0
$s2$	269	476	16	213	370	20	249	340	8	17
$s3$	63	60	6	57	71	8	74	93	4	5
$s4$	329	450	21	796	1,122	35	259	261	10	13
$s5$	194	181	25	139	80	25	124	67	13	8
$s6$	35	78	4	29	47	5	29	47	1	4
$s7$	204	688	13	81	155	12	81	145	1	13
$s8$	18	10	7	13	6	3	8	2	3	1
average	140	245	12	167	233	15	104	121	6	8

at every 20 nodes (= 104 seconds). Although the best frequency of the DC cut generation depends on instances, we experienced that generating DC cuts every 20 nodes works well, also.

Table 13: Frequencies of DC Cut Generation

Instance	Root node			Every 20			All nodes		
	time (sec.)	*B&C* nodes	DC cuts	time (sec.)	*B&C* nodes	DC cuts	time (sec.)	*B&C* nodes	DC cuts
$s1$	6	20	3	7	16	5	7	16	5
$s2$	893	2,300	8	249	340	25	343	387	37
$s3$	127	208	7	74	93	9	83	93	9
$s4$	10,604	24,590	9	259	261	23	464	392	44
$s5$	206	297	10	124	67	21	120	47	27
$s6$	40	135	3	29	47	5	29	39	6
$s7$	98	240	4	81	145	14	67	83	19
$s8$	8	2	4	8	2	4	8	2	4
average	1,498	3,474	6	104	121	13	140	132	19

11.1.3 Branching

We experimented with three branching strategies. The results are presented in Table 14. The results from the branching, with the priority order based on the tail node time, for each arc variable were somewhat counter to our expectation (the Cplex default branching strategy outperformed it). However, Table 14 clearly shows the effectiveness of the superarc branching.

Table 14: Comparison of Branching Strategies

Instance	Cplex default		Priority based on the tail node time		Superarc branching	
	time (sec.)	*B&C* nodes	time (sec.)	*B&C* nodes	time (sec.)	*B&C* nodes
$s1$	5	11	42	2,507	7	16
$s2$	951	6,706	(0.20%)*	(524,260)**	249	340
$s3$	50	123	204	990	74	93
$s4$	963	10,795	(10.81%)*	(176,680)**	259	261
$s5$	133	297	2,864	12,246	124	67
$s6$	26	220	22	164	29	47
$s7$	257	5,355	(3.27%)*	(849,640)**	81	145
$s8$	8	2	18	203	8	2
average	299	2,939			104	121

()* Gap after 10 hours.
()** # of *B&C* tree nodes after 10 hours.

11.2 Comparison of the RGH with the Time Discretized Model

The goal of the experiments in this section is to compare the RGH with the time discretized model. We solve 6 medium size instances both by RGH with 2000 iterations and by the time discretized model. The time unit used for the time discretized model is $(2, 4)$. The time discretized model may not always succeed to produce a better solution because of the time discretization. The results are presented in Table 15.

Table 15: Comparison of the RGH with the Time Discretized Model 1

Instance	RGH 2000	*B&C*			LP GAP	IP GAP
	TC_{RGH}	root node LP	IP	time (sec.)	(%)	(%)
m1	2,470.3	2,008.5	2,245.7	183,543	18.69%	9.09%
m2	1,852.8	1,590.1	1,840.0	47,068	14.18%	0.70%
m3	2,722.6	2,465.0	2784.5	124727	9.46%	-2.27%
m4	3,017.9	2,363.8	2,920.4	143,278	21.67%	3.23%
m5	3,107.1	2,681.6	2,970.5	523,286	13.69%	4.40%
m6	3,051.0	2,412.2	2,844.0	28,330	20.94%	6.78%
average	2,703.6	2,253.5	2,600.8	175,038.7	16.44%	3.65%

The column headed *IP GAP* (%) is computed as $\frac{TC_{RGH}-IP}{TC_{RGH}} \times 100$. The largest gap is 9.09%, produced in instance $m1$. RGH produced a better solution than the *B&C* for

instance $m3$. Note that the CPU time of the $B\&C$ is too large to use this method in practice. The CPU time of the RGH with 2000 iterations was less than 4 minutes for each instance.

The results in Table 15 show that the solution quality of RGH is reasonably good. The column headed *LP GAP* (%) is computed as $\frac{TC_{RGH} - \textit{root node LP}}{TC_{RGH}} \times 100$. In Table 16, we present *LP GAP* (%) with large instances (We derived these new instances from the base instance used in Chapter 8). These instances involve 7 production facilities, about 200 customers, and a homogeneous fleet of 7 vehicles. Since the difference in *LP GAP* (%) between medium size instances (Table 15) and large size instances (Table 16) is not significant , we can expect that the RGH works reasonably well even with large instances.

Table 16: Comparison of the RGH with the Time Discretized Model 2

	RGH 2000	$B\&C$	LP GAP
Instance	TC_{RGH}	root node LP	(%)
ni1	5,541.0	4,136.7	25.34%
ni2	6,778.2	5,715.4	15.68%
ni3	6,861.6	5,275.3	23.12%
ni4	9,004.5	6,597.0	26.74%
ni5	7,257.7	6,102.7	15.91%
nl1	7,123.7	5,341.1	25.02%
nl2	8,547.9	6,771.8	20.78%
nl3	8,294.3	6,638.5	19.96%
nl4	7,466.1	6,053.5	18.92%
nl5	6,333.4	5,659.7	10.64%
average	7,320.8	5,829.2	20.21%

CHAPTER XII

INTEGRATED APPROACH

In this final chapter, we outline an integrated approach using heuristics and optimization algorithms providing effective and efficient technology for solving inventory problems with continuous moves. Although we developed network reduction techniques and specialized IP techniques to solve the time-discretized model, it is still impossible to solve real size problems in a reasonable amount of time (see Chapter 11). The way we can use the developed solution methods in practice is by combining it with the randomized greedy heuristic. We propose an idea of how to use the time-discretized model to improve solutions obtained by RGH. The idea is simple. We decompose the problem into several small problems based on the solution obtained by RGH and solve the portions with the branch-and-cut algorithm. We can still use DVO to optimize delivery quantities after finding better solutions.

Suppose a solution from RGH is given. Based on the solution, we denote by $\mathcal{C}^k$ and $\mathcal{P}^k$, a set of customers and a set of plants visited by vehicle k, respectively. If each customer in $\mathcal{C}^k$ and each plant in $\mathcal{P}^k$ is visited by vehicle k only, then we can easily extract a small time-discretized model, which optimizes a part of the solution by solving a single vehicle problem with a set of customers $\mathcal{C}^k$ and a set of plants $\mathcal{P}^k$.

Even though a plant or a customer is visited by more than one vehicle, we can still handle this situation. We focus on extracting a single vehicle problem for vehicle k. Suppose plant $j \in \mathcal{P}^k$ is also visited by vehicle k'. Since we need to maintain solutions for the other vehicles, plant j has to hold enough product for the scheduled pickup by vehicle k'. Suppose vehicle k' visits plant j at time t to pickup $d^{k'}$ amount of product based on the solution. By simply using $I_{(j,t-1)} - d_{(j,t)} + p_{(j,t)} - d^{k'} = I_{(j,t)}$ instead of using $I_{(j,t-1)} - d_{(j,t)} + p_{(j,t)} = I_{(j,t)}$, we can guarantee that vehicle k' can pickup $d^{k'}$ amount of product at plant j at time t. Similarly, in the case of a customer visit, we use $I_{(i,t-1)} + d_{(i,t)} - u_{(i,t)} + d^{k'} = I_{(i,t)}$ instead of using $I_{(i,t-1)} + d_{(i,t)} - u_{(i,t)} = I_{(i,t)}$ in order to guarantee that vehicle k' can delivery $d^{k'}$ amount

of product to customer i. In this case, we need to compute delivery periods for customer i while considering outside delivery $d^{k'}$.

We start with $k = 1$. After solving the time-discretized model for vehicle k, if the time-discretized model found a better solution, we update the solution (delivery/pickup time and quantities) . We move on to the problem for vehicle $k + 1$. In the following computational results chapter, we give a simple example to show the effectiveness of this method.

The chance to find a better solution is increased by solving a time-discretized model for 2 vehicles as the improvement step. The improvement time-discretized model for vehicle k and k' includes a set of customers $\mathcal{C}^k \bigcup \mathcal{C}^{k'}$, a set of plants $\mathcal{P}^k \bigcup \mathcal{P}^{k'}$, and vehicle k and k'. We can use this approach in practice only when the improvement time-discretized model is solvable in a reasonable amount of time.

Table 17 shows an example of using the time-discretized model as an improvement step of RGH. We applied the method proposed in this chapter to instance $ni1$.

Table 17: Practical Use of the Time-Discretized Model

	TC_{RGH}	TC_{TD}	Best
vehicle 1	1,147.2	1,066.3	1,066.3
vehicle 2	751.8	520.3	520.3
vehicle 3	259.1	259.1	259.1
vehicle 4	788.2	infeasible	788.2
vehicle 5	1,336.1	infeasible	1,336.1
vehicle 6	615.5	640.5	615.5
vehicle 7	643.0	995.5	643.0
total	5,540.9		5,228.5

The column headed TC_{RGH} gives each vehicle's transportation cost from the solution obtained by RGH. The column headed TC_{TD} gives the optimal objective value of the improvement time-discretized model for each vehicle. The CPU time of the improvement time-discretized model was less than 10 minutes for each vehicle's problem. The improvement time-discretized model found better solutions for vehicle 1 and 2. For vehicle 3, TC_{RGH} and TC_{TD} are the same. The improvement time-discretized models for vehicle 4 and 5 are infeasible. For vehicle 6 and 7, the improvement time-discretized models failed to find a better solution. The total transportation cost is decreased from 5,540.9 to 5,228.5

by using the time-discretized model as an improvement step. The percentage improvement is 5.64%.

CHAPTER XIII

CONCLUSIONS

This thesis consists of two parts: "Performance Measurement for Inventory Routing" and "Inventory Routing with Continuous Moves". In Part I, we have developed a technology that allows a vendor to measure the effectiveness of its distribution strategy in an absolute sense. This technology can also be used to suggest desirable delivery patterns and to analyze tactical and strategic decisions. In Part II, we developed a technology that constructs distribution plans in environments where continuous moves are employed. The IRP-CM, the problem studied in Part II, captures production as well as consumption complexities. In this chapter, we give a brief summary of the results we have presented, the insight we obtained, and suggest a few directions for future research.

In Chapter 2, we presented a simple bound on the minimum total mileage required to satisfy customer demand. Subsequently, we analyzed two 2-customer examples. The analysis revealed a crucial insight, which forms the basis for the theorem that allows the removal of one of two obstacles to using the pattern selection LP; the number of feasible delivery patterns is prohibitively large. The optimal objective function value of the pattern selection LP provides a lower bound on the total mileage required to satisfy customer demand, and we proved that base patterns are sufficient to find an optimal solution to the pattern selection LP. The other obstacle to using the pattern selection LP is that the calculation of the cost of each delivery pattern involves the solution of a traveling salesman problem. Through the limitation of the number of customers visited in a delivery pattern, the second obstacle could be removed. However, we were able to obtain only lower and upper bounds on the optimal objective function value of the pattern selection LP.

In Chapter 3, we presented a variety of computational experiments conducted with real-life data to evaluate our proposed methodology. We observed that by limiting the number of customers visited in a delivery pattern to 3 or 4, we could obtain lower and upper bounds

close to the optimal objective function value of the pattern selection LP.

Finally, in Chapter 4 we discussed other potential uses of the technology developed. Through solving the small instance introduced by Fisher, we showed that the technology developed can suggest effective practical delivery trips as well.

There are several directions for future research related to the technology developed in Part I. As indicated, we have not considered time information of delivery patterns in the pattern selection LP. Thus the optimal objective function value of the pattern selection LP is a lower bound on the minimum total mileage required to satisfy customer demand. We may obtain tighter lower bounds through consideration of timing deliveries.

Additionally, we can develop decision support tools for strategic and/or tactical decisions since our work provides useful methodology for performance measurement. These decisions include facility location problems, customer-plant alignments, and capital investments to improve the efficiency of its distribution system.

In Chapter 5, we formally introduced the inventory routing problem with continuous moves in which a vehicle is allowed to visit any plant and any customer. In Chapter 6, we gradually developed the randomized greedy heuristic proposed for its solution. Although the main ideas and concepts used are similar to those of insertion heuristics for vehicle routing and scheduling problems, their actual implementation is more complicated due to the complex nature of the IRP-CM. Through the introduction of randomization into the heuristic, we improved the performance of the heuristic.

In Chapter 7, we presented the delivery volume optimization model used as a postprocessing step in the heuristic. The DVO LP takes a global look at the schedule produced by the randomized greedy heuristic and maximizes the total volume of product delivered over the planning period, by removing the minimum delivery requirement and reevaluating the times of visits and the amounts of product picked up or delivered.

In Chapter 8, we presented a variety of computational experiments of the heuristic and the delivery volume optimization conducted with data sets derived from real-life Argon production and distribution information. We presented computational results that demonstrate the benefits of our flexible insertion mechanism and of randomization. We were able

to show that the use of delivery volume optimization results in a higher volume delivered per mile for long planning periods.

In Chapter 9, we proposed a time-discretized model for the IRP-CM to solve to optimality. In Chapter 10, we discussed various solution techniques including problem size reductions and specialized IP techniques for the time-discretized model. The specialized IP techniques include delivery cover inequalities and superarc branching. We also developed two separation algorithms to determine delivery cover inequalities violated by the LP solutions. We also discussed how we can use these optimization techniques to improve the solutions from the randomized greedy heuristic.

In Chapter 11 we presented a variety of computational experiments conducted in order to see the effectiveness of the solution technologies developed. We also presented experiments conducted to compare solutions of the heuristic with those of the time-discretized model.

To develop even more effective approaches, we plan to focus on:

- Develop effective decomposition schemes based on the heuristic solution and then solve the portions with an optimization method, and
- Develop more effective/efficient optimization.

We have developed two separation heuristics to determine delivery cover cuts violated by the optimal LP solution. However, both of them may fail to detect existing DC cuts violated by the LP solution. Through devising different separation schemes, we can increase the chance of determining DC cuts violated, and this may yield a more effective $B\&C$ algorithm for the time-discretized model of IRP-CM.

Defining new sets of valid inequalities to tighten the LP relaxation of the time-discretized model is a good future research topic. Due to the similarity between the IRP-CM and the capacitated vehicle routing problem (CVRP), studying applicability of valid inequalities for the CVRP to the IRP-CM may be a good start position.

We already introduced a new branching rule in the further issue subsection in the discussion of branching rules. Branching on constraints which are related to the delivery cover inequalities. The other possibility is a variant of our current superarc branching. Instead

of branching on a superarc of which value is close to 0.5, we may branch on a superarc with a higher value, for example 0.7, because setting a superarc with 0.5 to 0 may have less impacts on the setting other variables than setting it to 1. These branching rules need to be experimented in order to evaluate their performance.

We want to study theoretical properties of delivery cover cuts. As mentioned before, delivery cover cuts are very similar to capacity constraints of the CVRP. Since every capacity constraint of the CVRP does not always define a facet of the CVRP polytope (see Naddef and Rinaldi [29]), it is not likely for every delivery cover inequality to define a facet of the IRP-CM polytope. However, under certain conditions, a delivery cover inequality may define a facet of the IRP-CM. If we can determine these conditions, we can use them to design more powerful separation routines.

REFERENCES

[1] Achuthan, N., Caccetta, L., and Hill, S., "An improved branch-and-cut algorithm for the capacitated vehicle routing problem," *Transportation Science*, vol. 37, pp. 153–169, May 2003.

[2] Adelman, D., "Price-directed replenishment of subsets: Methodology and its application to inventory routing," *Manufacturing & Service Operations Management*, vol. 5, pp. 348–371, October 2003.

[3] Adelman, D., "A price-directed approach to stochastic inventory/routing," *Operations Research*, 2004. To appear.

[4] Anbil, R., Tanga, R., and Johnson, E., "A global optimization approach to crew scheduling," *IBM Systems Journal*, vol. 31, pp. 71–78, 1991.

[5] Anily, S. and Federgruen, A., "One warehouse multiple retailer systems with vehicle routing costs," *Management Science*, vol. 36, no. 1, pp. 92–114, 1990.

[6] Anily, S. and Federgruen, A., "Rejoinder to 'one warehouse multiple retailer systems with vehicle routing costs'," *Management Science*, vol. 37, no. 11, pp. 1497–1499, 1991.

[7] Augerat, P., Belenguer, J., Benavent, E., Corberán, A., Naddef, D., and Rinaldi, G., "Computational results with a branch and cut code for the capacitated vehicle routing problem," technical report rr 949-m, Université Joseph Fourier, Grenoble, France, 1995.

[8] Bard, J., Huang, L., Jaillet, P., and Dror, M., "A decomposition approach to the inventory routing problem with satellite facilities," *Transportation Science*, vol. 32, pp. 189–203, 1998.

[9] Bell, W., Dalberto, L., Fisher, M., Greenfield, A., Jaikumar, R., Kedia, P., Mack, R., and Prutzman, P., "Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer," *Interfaces*, vol. 13, no. 6, pp. 4–23, 1983.

[10] Campbell, A., Clarke, L., and Savelsbergh, M., "Inventory routing in practice," in *The Vehicle Routing Problem* (Toth, P. and Vigo, D., eds.), pp. 309–330, SIAM Monographs on Discrete Mathematics and Applications, 2002.

[11] Campbell, A. M., *Inventory Routing*. PhD thesis, Georgia Institute of Technology, Department of Industrial and Systems Engineering, August 2000.

[12] Campbell, A. M., Clarke, L., Kleywegt, A., and Savelsbergh, M. W. P., "Inventory routing," in *Fleet Management and Logistics* (Crainic, T. and Laporte, G., eds.), Kluwer Academic Publishers, 1998.

[13] Campbell, A. M. and Savelsbergh, M. W. P., "Delivery volume optimization," *Transportation Science*, vol. 38, pp. 210–223, May 2004.

[14] Chan, L., Federgruen, A., and Simchi-Levi, D., "Probabilistic analyses and practical algorithms for inventory-routing models," *Operations Research*, vol. 46, no. 1, pp. 96–106, 1998.

[15] Chien, T., Balakrishnan, A., and Wong, R., "An integrated inventory allocation and vehicle routing problem," *Transportation Science*, vol. 23, no. 2, pp. 67–76, 1989.

[16] Christiansen, M., "Decomposition of a combined inventory and time constrained ship routing problem," *Transportation Science*, vol. 33, no. 1, pp. 3–16, 1999.

[17] Christiansen, M., Fagerholt, K., and Ronen, D., "Ship routing and scheduling: Status and perspectives," *Transportation Science*, vol. 38, no. 1, pp. 1–18, 2004.

[18] Dror, M. and Ball, M., "Inventory/routing: Reduction from an annual to a short period problem," *Naval Research Logistics Quarterly*, vol. 34, no. 6, pp. 891–905, 1987.

[19] Dror, M., Ball, M., and Golden, B., "A computational comparison of algorithms for the inventory routing problem," *Annals of Operations Research*, vol. 4, no. 1-4, pp. 3–23, 1985.

[20] Federgruen, A. and Zipkin, P., "A combined vehicle routing and inventory allocation problem," *Operations Research*, vol. 32, no. 5, pp. 1019–1036, 1984.

[21] Fisher, M., Greenfield, A., Jaikumar, R., and Kedia, P., "Real-time scheduling of a bulk delivery fleet: Practical application of lagrangean relaxation," tech. rep., The Wharton School, University of Pennsylvania, Department of Decision Sciences, October 1982.

[22] Gallego, G. and Simchi-Levi, D., "On the effectiveness of direct shipping strategy for the one-warehouse multi-retailer r-systems," *Management Science*, vol. 36, no. 2, pp. 240–243, 1990.

[23] Golden, B., Assad, A., and Dahl, R., "Analysis of a large scale vehicle routing problem with an inventory component," *Large Scale Systems*, vol. 7, no. 2-3, pp. 181–190, 1984.

[24] Jaillet, P., Bard, J., Huang, L., and Dror, M., "Delivery cost approximations for inventory routing problems in a rolling horizon framework," *Transportation Science*, vol. 3, pp. 292–300, 2002.

[25] Kleywegt, A., Nori, V., and Savelsbergh, M., "The stochastic inventory routing problem with direct deliveries," *Transportation Science*, vol. 36, pp. 94–118, 2002.

[26] Kleywegt, A., Nori, V., and Savelsbergh, M., "Dynamic programming approximations for a stochastic inventory routing problem," *Transportation Science*, vol. 38, pp. 42–70, 2004.

[27] Lysgaard, J., Letchford, A., and Eglese, R., "A new branch-and-cut algorithm for the capacitated vehicle routing problem," *Mathematical Programming*, vol. 100, pp. 423–445, June 2004.

[28] Minkoff, A., "A markov decision model and decomposition heuristic for dynamic vehicle dispatching," *Operations Research*, vol. 41, pp. 77–90, 1993.

[29] Naddef, D. and Rinaldi, G., "Branch-and-cut algorithms for the capacitated vrp," in *The Vehicle Routing Problem* (Toth, P. and Vigo, D., eds.), SIAM Monographs on Discrete Mathematics and Applications, 2002.

[30] Nori, V., *Algorithms for Dynamic and Stochastic Logistics Problems*. PhD thesis, Georgia Institute of Technology, Department of Industrial and Systems Engineering, November 1999.

[31] Ralphs, T., Kopman, L., Pulleyblank, W., and Trotter, L., "On the capacitated vehicle routing problem," *Mathematical Programming*, vol. 94, 2003.

[32] Resende, M., "Greedy randomized adaptive search procedure (grasp)," technical report, AT&T Labs Research, 1998.

[33] Webb, R. and Larson, R., "Period and phase of customer replenishment: A new approach to the strategic inventory/routing problem," *European Journal of Operational Research*, vol. 85, no. 1, pp. 132–148, 1995.

VITA

Jin-Hwa Song was born in Anyang, Korea on October 24, 1971. In the spring of 1990, he entered Korea University where he received a Bachelors degree in Industrial Engineering in 1996 and a Master of Science in Operations Research and Systems Engineering in 1998. In the fall of 1998, Jin-Hwa left for Atlanta to continue his education at the Georgia Institute of Technology in the School of Industrial and Systems Engineering. At Georgia Tech, he was the Praxair scholar. In August of 1999, he earned a Master of Science in Operations Research, and in July of 2004, he was awarded the Doctor of Philosophy in Industrial Engineering.